RECONFIGURABLE G & C COMPUTER STUDY FOR SPACE STATION USE

FINAL REPORT

VOLUME II

FINAL TECHNICAL REPORT

31 January 1971

J. Jurison
Program Manager

Prepared Under Contract No. NAS9-10416

Autonetics Division of North American Rockwell
for
Manned Spacecraft Center
National Aeronautics and Space Administration

FOREWORD

This final report covers the work performed by Autonetics Division of North American Rockwell Corporation under a study contract entitled Reconfigurable G&C computer Study for Space Station Use. The report is submitted to the National Aeronautics and Space Administration Manned Spacecraft Center under the requirements of Contract NAS 9-10416. The study program covered the period from December 29, 1969 through January 31, 1971. The NASA Technical Monitor was Mr. E. S. Chevers.

The final report consists of seven (7) volumes:

Volume I	Technical Summary
➧Volume II	Final Technical Report
Volume III	Appendix I. Model Specification
Volume IV	Appendix 2. IOP - VCS Detailed Design
Volume V	Appendix 3. System Analysis and Trade-Offs
Volume VI	Appendix 4. Software and Simulation Description and Results
Volume VII	Appendix 5. D-200 Computer Family
	Appendix 6. System Error Analysis
	Appendix 7. Reliability Derivation for Candidate Computers
	Appendix 8. Power Converter Design Data
	Appendix 9. Data Transmission Medium Design

➧YOU ARE READING THIS VOLUME

VOLUME II
TABLE OF CONTENTS

VOLUME II
CONTENTS (Cont)

VOLUME II
CONTENTS (Cont)

VOLUME II
LIST OF ILLUSTRATIONS

VOLUME II
ILLUSTRATIONS (Cont)

VOLUME II
LIST OF TABLES

1.0 INTRODUCTION

1.1 SUMMARY

The purpose of the study is to define the general system requirements and specify the configuration for a modularized reconfigurable, fault tolerant guidance and control computer system suitable for a manned space station complex. The study includes power distribution, modular computing elements at various subsystems, input/output data bus, and the development of necessary software to demonstrate the self-test and reconfiguration ability of the system by computer simulation. This volume contains a summary of the results of the study.

1.2 GENERAL REQUIREMENTS

The computer system is required to support the Guidance and Control (G&C) requirements of the Space Station during each mission phase. The Space Station is expected to operate in a circular 200 - 300 mile orbit of 55 degree inclination with added capability for polar orbits. The mission can be broken down into four major phases: Prelaunch, Boost, Orbit Injection and Orbital Coast.

The orbital coast phase was the phase of primary concern for this study and is used to estimate the memory size, speed, and signal interface requirements for the computer system. The G&C system consists of several subsystems as shown in the functional block diagram, Figure 1-1.

The G&C computer and computing elements at subsystem level perform all computational tasks associated with navigation and attitude control function. Data processing functions associated with the experiments and display and control functions are handled by another computer complex, called the Information Management Data Processor. The latter provides mode control signals and receives navigation data from the G&C computer.

A common multiplexed data bus provides a means of transferring data between the subsystems and the guidance and control computer complex.

The manned environment and long periods of independent operation dictate more stringent reliability requirements than have been imposed upon spacecraft computers in the past. It has to be modular for ease of maintenance and be tolerant to three failures in a fail op, fail op, fail safe manner. This means that it must be able to detect failures, isolate failures to a modular level, and recover from failures by reconfiguring the system (replacing the faulty module with a spare).

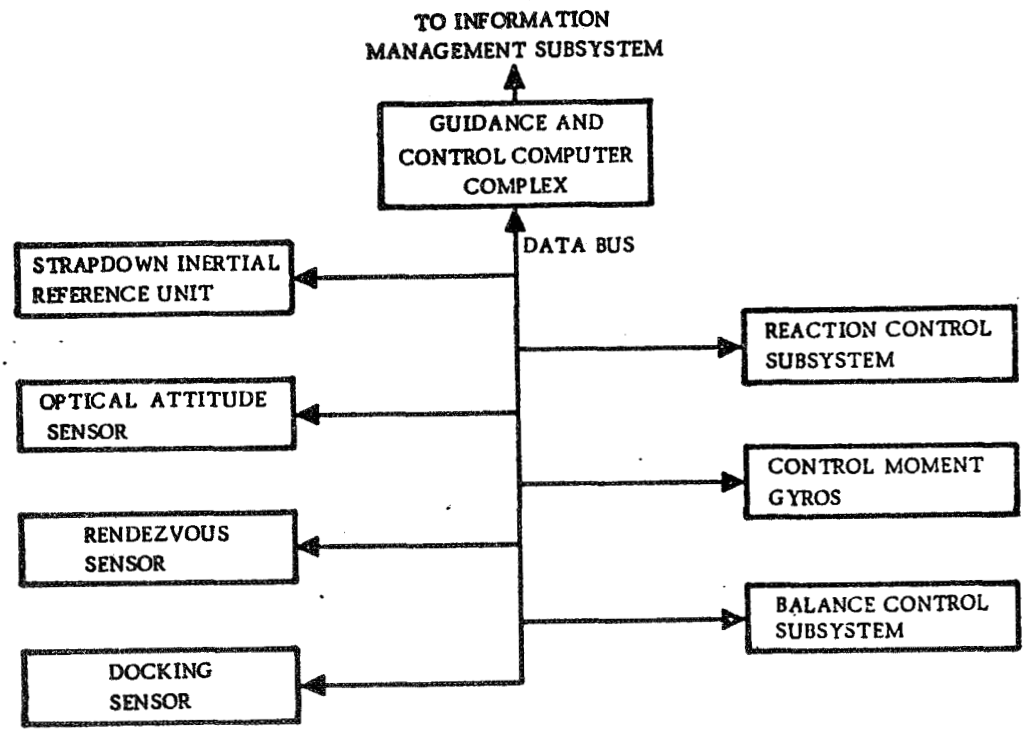


FIGURE 1-1. FUNCTIONAL BLOCK DIAGRAM , G&C SYSTEM

1.3 PROGRAM PLAN

The study has been organized into individual tasks as shown in Figure 1-2. The task descriptions are described in the Detailed Program Plan (Reference 1-1). It should be noted that the individual tasks are closely interrelated and somewhat iterative in nature. Task 12, Cost and Schedule Plan for Breadboard System Development, has been deleted by direction of NASA.

This volume presents the objectives and results of each of the tasks in detail. In cases where the supporting data is too voluminous, it has been included in the appendices.

This volume consists of eleven (11) sections. This section presents the background and overall objectives and requirements of the study. Section 2 presents the results of the technology review and defines the recommended logic circuit, memory and packaging technologies. Section 3 summarizes the system requirements, computer requirements, and includes the trade-off data that lead to the overall system concept. Section 4 describes the failure tolerance requirements and their impact on the selection of candidate systems. Section 5 presents the evaluation model developed for the candidate systems and summarizes the results of the evaluation. Section 6 is devoted to the technical discussion on the I/O data bus. Section 7 describes the recommended Reconfigurable G&C Computer System mechanization in detail. Section 8 describes the software developed during the study and summarizes the results of the simulation. Section 9 presents the results of the Local Processor (LP) trade-offs and describes the recommended LP design. Section 10 presents the power distribution study results and Section 11 presents the recommendations for future effort.

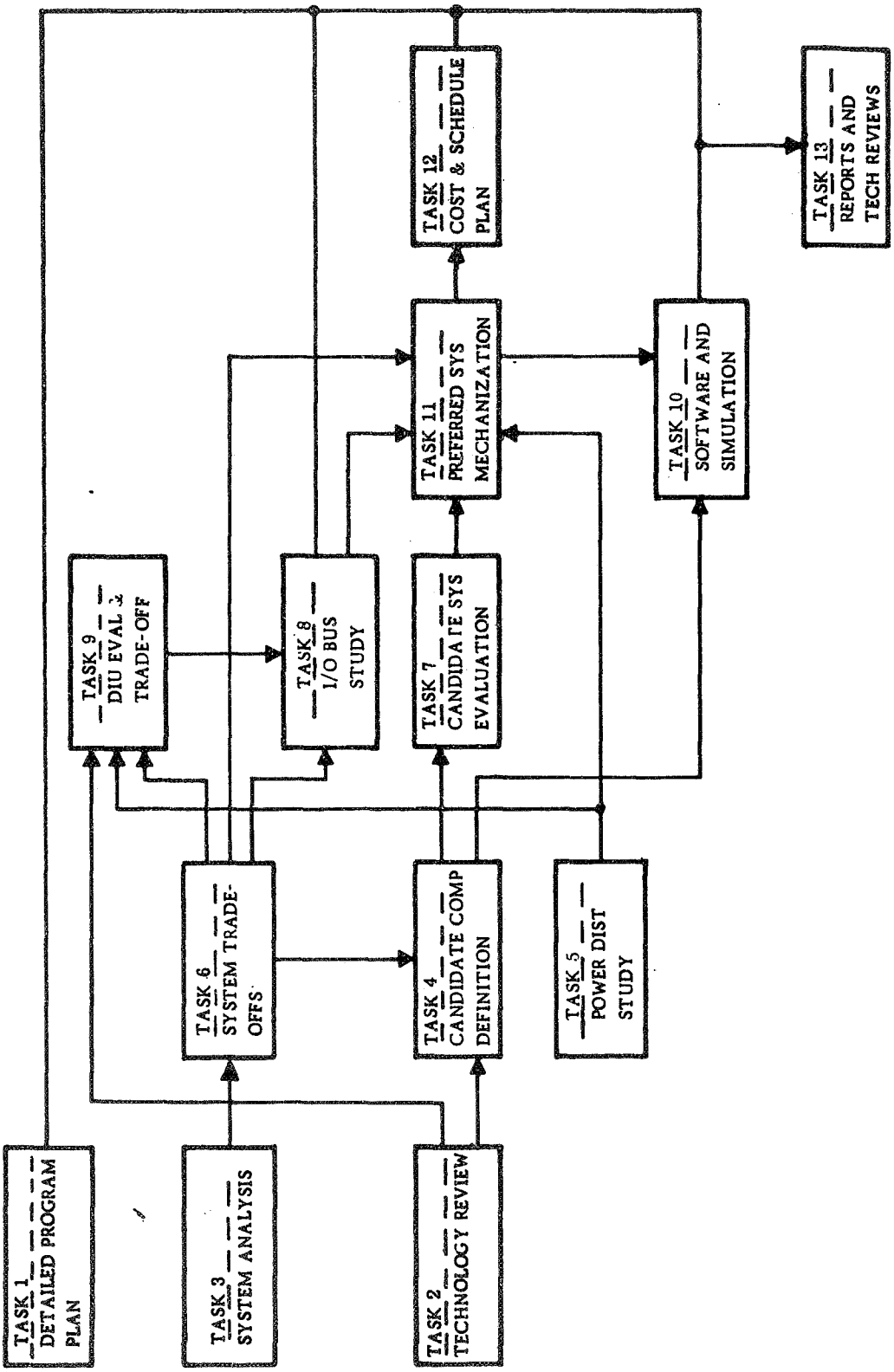


Figure 1-2. Program Flow

2.0 TECHNOLOGY REVIEW

2.1 INTRODUCTION

In order to establish a technological baseline for system tradeoffs in the Reconfigurable G&C Computer Study, a brief technology review has been conducted. The review covered three areas: semiconductor logic technologies, memory technologies (magnetic and semiconductor), and packaging. General ground rules for the review were that technologies considered should be producible in 1972 and that no substantial development should be required. Reliability was considered the most important factor in evaluating a technology but other characteristics important to spaceborne systems such as size, weight and power were also covered. Ground rules for particular technological areas are given below.

2.1.1 Semiconductor Logic

Logic bit rates of typically 1 Mhz are expected in the Reconfigurable G&C Computer system. Logic technologies were limited primarily to four phase P-channel MOS, complementary MOS and low power bipolar including bipolar MSI.

2.1.2 Memories

Memory modules of interest in this system are likely to be from 2 to 16K words of from 24 to 36 bits each. Parity would be included. The modules may require multi-access ports for multiprocessor configurations. Cycle times will probably not be less than 1 μsec. with 0.5 μsec. access. The bulk of the effort was concentrated on core, plated wire, and films for magnetic memories and dynamic P-channel MOS, complementary MOS, and bipolars for semiconductor memories.

2.1.3 Packaging

From reliability considerations this effort was limited to characterizing various methods of packaging uncased devices on ceramic substrates.

2.2 SEMICONDUCTOR LOGIC TECHNOLOGY

2.2.1 General

Semiconductor technologies under consideration for use in logic portions of the Reconfigurable Computer have been limited to those with proven reliability and producibility. Three technologies, bipolar, P-channel MOS, and complementary MOS (CMOS) fall into this category. Both bipolar and P-channel circuits are currently being produced by a large number of

2.2.1 (continued) manufacturers and therefore represent virtually zero risk. CMOS, because of the limited number of suppliers, must be considered as a slightly higher risk.

The reliability of a system can be greatly enhanced by reducing the number of connections required. One method of accomplishing this is to increase the level of integration on the semiconductor devices to reduce interface signals. Care must be taken however, in the logical apportionment of the system or the number of signals can actually increase. Of the three technologies being considered, P-channel MOS is capable of the highest level of integration, approximately twice that achievable with either bipolar or CMOS technologies. There is no reason to expect that this ratio is likely to change in the future, although higher levels of integration will be achieved in all technologies.

Another method of reducing system interconnects is through the use of uncased devices on ceramic interconnecting substrates. These interconnection techniques are applicable to all the semiconductor technologies under consideration. They eliminate one or two of the three connections required by conventional packaging techniques.

The inherent reliability of semiconductor technologies can be related to the number of steps in the fabrication process and the criticality of masking steps. Control of the gate oxide thickness is usually considered the one critical step in MOS circuit fabrication. The gate oxide thickness, however, can be determined visually unlike the critical diffusion depths in bipolars. Furthermore, the addition of silicon nitride to the gate insulation allows a thicker gate insulation of less critical thickness. The thicker gate insulation also greatly reduces the possibility of device failures due to gate insulation shorts.

Power is an important reliability factor in integrated circuits. Many failure modes are accelerated by high temperatures. It is desirable to keep chip temperatures low by using low power circuit techniques. Both 4-phase P-channel MOS and CMOS have very low power dissipations. Bipolars generally have high power dissipations, e.g., a large array may dissipate 1 watt of power. The requirement to remove large amounts of heat from the chips may limit use of high reliability interconnection schemes with uncased devices. Since heat must be conducted through smaller interconnects in order to cool the chip, the amount of heat that can be dissipated on a ceramic substrate is limited to 2 to 4 watts. No more than 4 large bipolar arrays could, therefore, be interconnected per substrate.

2.2.1 (continued)

It is concluded that P-channel MOS is the most suitable semiconductor logic technology for the Reconfigurable G&C Computer. The technology is well established and is capable of the highest levels of integration. It is compatible with high reliability interconnect techniques and is low power when a 4-phase clocking circuit mechanization is used. This permits high density packing which minimizes both size and weight of the system.

2.2.2 Bipolar Technology

Bipolar integrated circuits have evolved to a point where their performance, cost and reliability make them prime candidates for virtually any system. The wide variety of types and functions available allow for ease and speed of system design. Most types of bipolar logic have low power versions available for applications where high speed performance is not required. These begin to compete from the power standpoint with MOS logic at speeds around 1 Mhz. Bipolar technology has been somewhat slow in the introduction of devices of higher complexities. More medium complexity devices, however, have become available in the past year.

A number of bipolar circuit families are available for consideration including DTL, ECL, RTL, and TTL. ECL and RTL can be eliminated for the following reasons. Preliminary systems specifications indicate that system reliability is of primary importance and that only moderate circuit speeds are required. Power is a consideration although of secondary importance. ECL is very fast and has a good speed/power product. This, however, is only realized when operating at maximum speed. Since only moderate speeds are required, the higher power of ECL over other circuit types is not justified. RTL tends to be too slow for this application and has limited fanout capability. Both ECL and RTL have low noise immunity which could lead to transient system failures. Neither ECL or RTL has a significant number of MSI devices available. MSI or LSI devices can increase reliability by reducing interconnections.

Both DTL and TTL are definite candidates for the reconfigurable computer. Almost every major semiconductor manufacturer makes a line of DTL or TTL or both. The two types are electrically compatible so that they may be mixed if desired in a system. Both types have low and medium speed (power) lines available. A high speed TTL line is also available but it is faster than is required in this application and its higher power cannot be justified. The medium speed TTL has a typical gate delay of 10 nsec, DTL has a gate delay of 25 nsec and low power TTL a delay of 33 nsec. The compatibility of these circuits allows regular TTL to be used where speed or drive is required and DTL or low power TTL to be used in less critical areas.

2.2.2 (continued)

The TTL lines have the most extensive number of logic functions available. These range from single flip-flops and quad gates to MSI devices. A brief summary of the types of functions presently available is given below:

Function	Type	Delay/ Freq.	Power (Avg.)	Remarks
Quad 2 input NAND	TTL	10 nsec	40 mw	typical
Quad 2 input NAND	DTL	20 nsec	72 mw	
Quad 2 input NAND	LPTTL	33 nsec	1.8 mw	
Dual J-K F/F	TTL	20 Mhz	100 mw	
	LPTTL	3 Mhz	7.2 mw	
4-bit counter	TTL	18 Mhz	128 mw	typical
4-bit counter	LPTTL	3 Mhz	16 mw	typical
4-bit Full Adder	TTL	60 nsec	390 mw	max delay
4-bit Arith Logic	TTL	28 nsec		
8-bit Shift Reg.	TTL	18 Mhz	175 mw	typical
8-bit Shift Reg.	LPTTL	6.5 Mhz	17.5 mw	typical
BCD to Decimal Decoder	TTL	25 nsec	140 mw	typical

In the event that standard MSI devices are not available, desired functions can be mechanized with "discrete" integrated circuits, or custom MSI devices can be designed. Completely custom bipolar designs are expensive and require a long lead time, but this problem can be largely circumvented by the use of standard cell arrays.

The standard cell array approach configures a standard matrix of gates into many different functions by a custom metal interconnection pattern. Two configuration techniques, discretionary wiring and fixed array, are currently available.

2.2.2 (continued)

Discretionary wiring involves fabricating a wafer of cells, determining the good cells by testing them individually, and then using a computer program to generate a unique metallization pattern for each wafer using only the good cells. The unique metallization pattern presents a reliability question since no two devices of the same function are likely to have the same interconnect pattern. Since the large wafer is packaged the size advantage of MSI/LSI is greatly reduced. This technique requires a two to three layer metallization which complicates the process and subsequently reduces the yield.

A more satisfactory approach is the fixed array. Here a standard array of cells is fabricated in which all cells are assumed good. A fixed custom interconnect pattern is designed to mechanize the desired function and is superimposed over the array. A large number of arrays are fabricated on each wafer and bad arrays are discarded through testing in the usual semiconductor method. Each array fabricated in this manner is identical and depending on yields, many arrays may come from each wafer. The chip size is comparable with other technologies and therefore similar packaging can be considered. At present, standard cell arrays of from 12 to 112 cells are available. The smaller arrays require two layers of metalization while the 112 gate array requires three. Power can be a serious problem in large bipolar arrays. A 112 cell array will dissipate approximately one watt and therefore must be provided with an adequate heat sink. Beam leading or flip chipping such devices is very questionable since all heat must be conducted through the leads. Power will also limit the number of devices that can be mounted on a common substrate.

2.2.3 Four-Phase P-Channel MOS

The P-channel MOS technology is a fully developed, mature technology applicable to a wide variety of systems. Numerous manufacturers are currently producing P-channel MOS circuits while new process techniques are coming into use which enhance both performance and reliability. Two circuit techniques are most commonly used in P-channel MOS: two-phase (2-ϕ) and four-phase (4-ϕ). Of the two, 4-ϕ circuits have significant advantages in both power and functional density.

Four-phase circuits require only that power dissipated in charging and discharging circuit capacitances. In this respect they are similar to CMOS and unlike 2-ϕ MOS or bipolar circuits which require power consuming resistive voltage dividers. At no time does a DC path exist between power supply and ground in a 4-ϕ gate. The average clock power for a 4-ϕ LSIC is about 50 mw at 1 Mhz. This power can be reduced in standby situations or for modules not requiring that high a frequency by reducing the clock frequency.

2.2.3 (continued)

The basic 4-ϕ gate (using P-channel devices) is shown in figure 2-1. The gate operation proceeds as follows:

1. During time T_1 and T_2 clock ϕ_1 is able to charge the gate capacity to a negative voltage.

2. During T_2 clock ϕ_2 is additionally able to charge the load capacity. At end of T_2 both the gate and load capacity are charge negative unconditionally. Thus, the output is not valid during T_2.

3. At T_3 clock ϕ_1 is zero so that one end of the gate is at ground and the device connecting the gate to the negative supply is off. If a path exists through the logic, the gate capacity will be discharged as well as the load. If no path exists, the load will remain negative.

4. At T_4 clock ϕ_2 is zero so the load is isolated from the gate. The output is valid until clock ϕ_2 connects the load with the gate.

5. In general, to perform logic, other clocks are mechanized to accept inputs at other times so that their operation overlaps this gate. Only 2 clock phases are needed for shift registers. Three phases are needed for general logic functions. Four phases are generally used to provide more efficient logic mechanizations.

The high functional density obtainable in MOS circuits results from a simple process and the ratioless nature of 4-ϕ logic. The single channel MOS process requires only one diffusion step. The FETS and diffused gate structures are self-isolating and therefore no chip area is required for isolation. Only one FET is required in a gate per logical input. The size of the FETS are determined only by the required gate switching speed and its output capacitive load. In many cases FETS can be the smallest producible size (0.6 x 0.6 mils). One or two extra FETS are required per gate, however, to perform the precharge and isolation functions.

Four-phase circuits have found applications ranging from aerospace digital computers to desk top calculators. A 24-bit parallel general purpose computer was developed by Autonetics in 1968-1969 which is mechanized with 8 MOS/LSI device types.

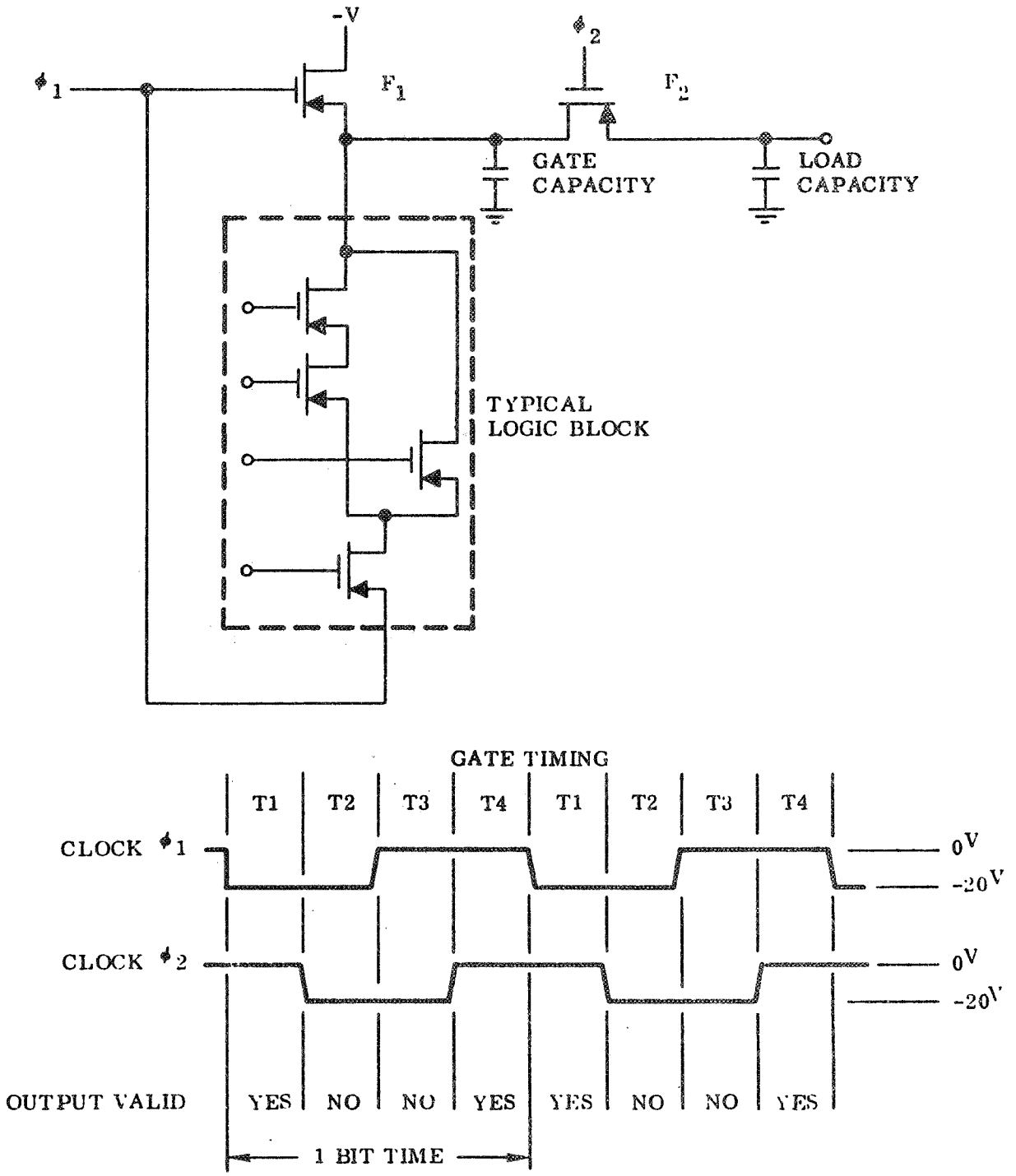


FIGURE 2-1. BASIC 4-PHASE GATE

2.2.3 (continued)

An advanced 4-ϕ system has been developed which enhances circuit speed, eliminates major inter-circuit noise and extends the logical capability available to the designer. Up to eight levels of inverting logic can be obtained in a single clock period. Non-inverting levels can be many more. Each gate has maximum flexibility in communicating with other gates. Some 4-ϕ systems put serious restrictions on the inter-communications between gates. Using the advanced clocking system, complex LSI devices have been developed with over 200 inverting logic gates mechanized with over 1500 FETS. The circuits operate at clock rates in excess of 1 Mhz (4 Mhz phase rate). A new family of computer building block devices is currently under development at Autonetics. A detailed description of these and other MOS/LSI devices is given in Appendix 5 where the use of such devices as building blocks is discussed.

In addition, it should be noted that considerable work is being done by several manufacturers on silicon gate MOS devices. This technology uses polycrystalline silicon for the gate electrode and offers a reduced threshold voltage on the gate. The significant result is the higher speed and bipolar compatibility that can be achieved with these devices. The maturing of this process will result in the mix of MOS and bipolar devices to use each to its fullest advantage.

2.2.4 Complementary MOS

Complementary MOS (CMOS) technology is beginning to emerge as a competitive technology to both bipolar and P-channel MOS. Until recently, CMOS costs have been extremely high and its use was limited to applications where its ultra-low standby power was of paramount importance. A limited number of standard functions, however, are now available at competitive prices. Most of these fall into the MSI category. One example is a 4-bit parallel adder with carry look ahead. Another is a 7-bit binary counter. More complex circuits are available on a custom basis. In the memory area, 64-bit memory chips are available and a 256-bit chip has been developed but is not now generally available.

The main reason for the slow development of CMOS appears to be the complex process required to fabricate the devices. The small number of suppliers certainly supports this contention. Only 3 or 4 semiconductor manufacturers are generally recognized as CMOS sources. Most of the major semiconductor companies, however, are currently engaged in process development for CMOS. Principal difficulties involve the necessity of isolating substrate areas of opposite polarities and the ability to control the threshold voltages of both the N and P-channel FETS simultaneously.

2.2.4 (continued)

The advantages of CMOS are its low standby (quiescent) power dissipation, relatively high speed and high noise immunity. These are offset by the more complex process and a lower functional density than is obtainable with some other technologies. Quiescent power dissipation of a typical CMOS MSI function is about 5 μw. Operational power will be considerably higher. Like all gating technologies, CMOS gates must switch their output capacitances. This represents a power loss proportional to the square of the supply voltage and the switching rate. The ripple-through nature of CMOS logic may also cause switching spikes which add to these losses through unnecessary switching. Furthermore, with the 15 volts required to achieve high switching speed, DC paths exist between the supply and ground during switchings. Substantial power can be dissipated in this manner, particularly if signal trnsitions are relatively slow. Gate delays in CMOS are about four times bipolar delays and half of P-channel delays. This assumes that gate structures are kept simple. Gates with large fan-in will be slower than equivalent P-channel gates.

The functional density of CMOS is approximately equal to that of current bipolar MSI but less than that achieved with P-channel MOS. One reason for this is the additional area required on the chip to isolate separate substrate areas for N and P-channel FETS. Another is the duality required in the classical complementary gate structure. Two MOS devices are required for each gate input; one N-channel and one P-channel. Any CMOS gate with more than two inputs will require more MOS FETS than an equivalent single channel gate. The FET sizes also tend to be larger than in single channel ratioless logic. The structure duality can be further illustrated by referring to the CMOS NOR gate in Figure 2-2. Note that while the P-channel devices are connected in parallel (ORed), the N-channel devices are in series (ANDed). A string of series devices is a speed limiting factor which cannot be avoided in classical CMOS logic.

2.2.5 Cross Technologies

By combining technologies in a single semiconductor device, some of the best characteristics of each can be realized. Among the potentially beneficial combinations are PMOS and bipolar, CMOS and bipolar, and CMOS and 4-Ø. One of the problems in PMOS is driving the interface capacitance between devices. By adding an N+ diffusion to the PMOS process, it is possible to produce both common collector NPN and lateral NPN bipolar transistors. The common collector NPN has high saturation resistance and the lateral NPN has low current gain. Nevertheless, the common collector NPN's can advantageously be used as emitter follower output drivers. These drivers have superior drive to straight PMOS drivers and load the internal circuitry less. The lateral NPN could be used to mechanize a current sense amplifier on a PMOS memory device. This could reduce access times to 100 nsec or less.

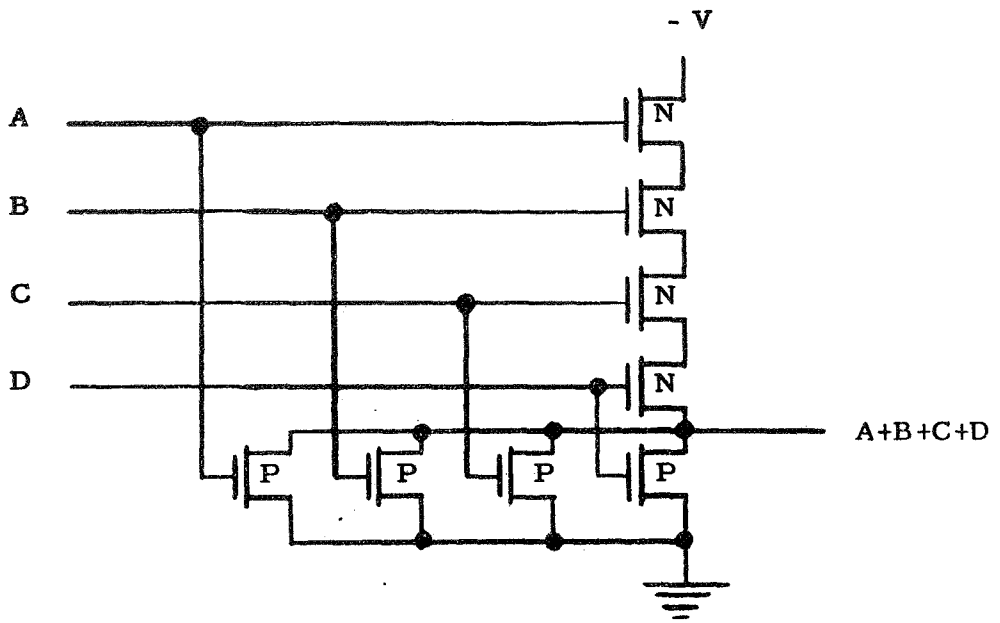


FIGURE 2-2. CMOS NOR GATE

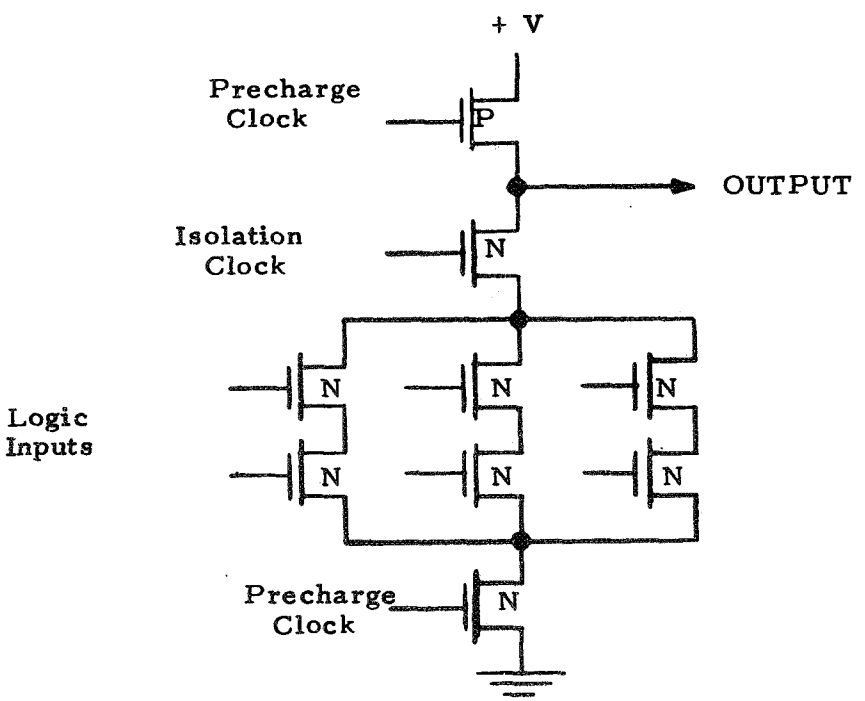


FIGURE 2-3. 4-PHASE CMOS GATE

2.2.5 (continued) The same bipolar transistors can be fabricated with CMOS. In this case, it is possible to obtain the devices with the existing diffusion steps.

Combining 4-Ø circuit techniques with a CMOS process can produce devices with the packing density of 4-Ø, speed equal or greater than conventional CMOS and power 1/2 to 1/4 that of 4-Ø PMOS. Figure 2-3 shows the circuit for a 4-Ø CMOS gate. Note that the circuit is mechanized with one device per logic input and therefore maintains the 4-Ø advantage in number of devices over conventional CMOS.

The precharge is the only P-channel device in the gate. The gate is precharged throughout this device by a clock going to ground. The threshold drop encountered in a 4-Ø PMOS gate is therefore eliminated. This permits a drop in supply voltages and clocks with a resulting decrease in power. Precharge speed is increased since the precharge device is operated in a grounded source configuration instead of as a source follower. The isolation and logic devices are all N-channel devices. The higher mobility of N-channel devices over P-channel may be traded off several ways. By maintaining logic levels at the same voltage and device sizes the same as in PMOS gates, an increase of up to three times PMOS speed can be obtained. If the logic levels are maintained but the logic devices made as small as possible, the chip size can be reduced or the functional density increased. Some limited speed increase may still result. Finally, if supply and clock voltages are decreased, speed equivalent to PMOS gate can be obtained at about 1/4 power. Conventional CMOS output drivers would improve capacitive drive and CMOS inverters could be used between 4-Ø gates as required.

2.3 MEMORY TECHNOLOGY

2.3.1 Semiconductor Memories

Semiconductor memories have been limited in the past to scratch pad applications where extremely high speed was required. Their low density, high power, and high costs barred them from other applications. Recent developments, particularly in the MOS area, have increased density and reduced both power and cost to a point where semiconductors are now replacing magnetics as main frame memories. This is especially true in small and medium size memories such as required in microprocessors and digital interface units. A summary of semiconductor memory device characteristics is given in Table 2-1. More detailed discussions of the various semiconductor technologies are presented in the following sections.

2.3.1.1 P-Channel MOS Memories - The highest density random access read/write semiconductor memories currently available are P-channel

2.3.1.1 (continued) MOS devices. Devices for 64 to 1024 bits are currently available or will be in the very near future. PMOS memories are of two types: static and dynamic. Two 256-bit static memory devices are currently on the market. One is completely self-contained including address decode and I/O circuits. It is mechanized on a small 110 x 122 mil chip using the silicon gate MOS process. Access time is about 1 μsec and the power is 2 mw/bit during access and 50 μw during standby. The other static device does not include full address decoding or I/O circuitry. While this requires external decode and sense devices, it permits memory systems with access times as fast as 100 nsec. Power per bit is 1 mw and this may be reduced substantially by strobing the power to the chip.

The power required for decode and sense must be added to the basic memory device power.

Dynamic memory devices differ from static devices in that they use the charge on a capacitor as the storage medium and require clocking at a minimum rate to retain information. Memories of 512 to 1024 bits per chip are pratical using this technique. Several manufacturers, including Autonetics, are currently developing devices of this type. Autonetics' device contains 512 bits and requires three clock signals. The device includes partial address decoding and complete I/O circuitry. Two configurations have been designed; a 512 x 1-bit chip and a 64 x 8-bit chip. Access times including full address decode are less than 500 nsec. The power is 200 μw/bit when clocked at 1 Mhz. Substantial reductions in this power are possible by gating the clocks during standby. A 4096 word 32-bit memory system using these devices can have a standby power less than 400 mw. Such a system could be constructed using conventional flat packs on from 4 to 8 printed circuit boards.

As an example of the use of the Autonetics 512 word by one bit device, a 4096 word by 4-bit memory requires 32 devices (RWMD 30024) and two address decoder (AD 30021) devices. Three sections of the two address decoders perform a decode of the first 9 bits of address to form the X, Y, and Z inputs to each RWMD device. The fourth section of the two decoders decode address bits 10, 11, and 12 to form a device select of 1 of 8 blocks of 4 RWMD devices. All input signals can have a logic "0" of from 0 to +2 volts and a logic "1" from -5 to -25 volts.

Addressing of the memory cells is accomplished by address bits 1, 2, and 3 forming the 1 of 8 (X) select lines, bits 4, 5, 6, forming the 1 of 8 (Y) select lines, bits 7, 8, 9, forming the 1 of 8 (Z) select lines. The X, Y, and Z select lines will activate one of the 512 cells in each device. The 1 of 8 device selects will activate a block of 4 RWMD devices forming a 4-bit read or write cycle.

TABLE 2-1. SEMICONDUCTOR MEMORY SUMMARY

Type	Technology	Bits	Access	Mw/Bit	Cents/Bit	Time
Read/Write	Bipolar	16-64	15-60 ns	5-10	30-300	Now
Read/Write	PMOS	256-1024	60-1000	-05-2	2-10	Now
Read/Write	CMOS	16-256	60-500	.005	100	Now
Read/Write *	MNOS	1024-?	0.5 μs	-	1	1973
Read Only	MOS	1024-4092	0.2-1.5 μs	-	1	Now

* Slow write

2.3.1.1 (continued)

With the address, read command and clocks applied, a read cycle will occur. The information data output from the memory will be available at the output in less than 500 nsecs after the application of an address code. The readout is non-destructive so there is no need for a rewrite cycle after read. With a write command, address and clocks applied, a write cycle will occur.

These 4096 word blocks can be connected together to form larger memories with the special gate in each address decoder used to select the block of memory desired.

The most convenient building blocks using the above described memory devices contain 2 address decoders and 32 memory devices. Combinations of these blocks can be configured into any desirable size memory system. Receiver drivers may be needed when loads exceed the device drive capabilities. The number of devices needed for various configurations is given in Table 2-2.

TABLE 2-2. MEMORY DEVICE REQUIREMENTS

Memory Size	No. of Address Decoders (AD 20031)	No. of 512 Word x 1-Bit Memory Dev. (RWMD.30024)	Total Bits
512 words x 32 bits	2	32	16,384
1024 words x 32 bits	3	64	32,768
4096 words x 32 bits	12	256	131,072
8192 words x 32 bits	24	512	262,144
16,384 x 32 bits	48	1024	524,288
32,768 x 32 bits	96	2048	1,048,576
64,536 x 32 bits	192	4096	2,097,152

2.3.1.2 CMOS Memories - It is in the area of memories that the low standby power of CMOS is most significant. Memories by nature are essentially in standby at all times. Only the small number of storage locations being accessed at any given time are active. The great majority of locations are in standby. If the power necessary to maintain the states of these cells can be minimized, substantial power saving can be realized. At the present time, CMOS offers the lowest standby power of all semiconductor memories.

2.3.1.2 (continued)

CMOS memory devices still suffer from lower packing density than is possible from single channel MOS. The largest CMOS memory device reported is a 256-bit array on a 17,400 sq. mil die. Twelve FETS are required for each bit. Access time is approximately 305 nsec.

The above 256 bit array is not generally available today but it is reasonable to expect CMOS memories of this density to be producible in 1972. A 16-bit chip has been available for several years. Sixty-four bit memories have been developed and at least one manufacturer is expected to market such a device.

2.3.1.3 Bipolar Memories - In applications requiring high speed memory access, bipolar semiconductor memory circuits may be required. Several manufacturers are currently producing 64-bit bipolar memory devices. The typical access times for these chips is 60 nsec. The penalty for this speed is power. Typical power dissipations are 5 to 6 mw/bit. Both conventional flat packages and beam lead chips are available. For extremely fast memories, 64-bit devices having access times of 5 nsec and requiring 10 mw/bit are available

2.3.1.4 MOS Read Only Memories (ROM) - Very compact ROM's can be produced using the MOS technology. 4096 bit ROM's are currently available. MOS ROM arrays are very similar to diode arrays. The presence or absence of a MOS device or a diode denote whether a one or a zero is stored. A single bit of storage can occupy as little as 1.5 mils^2.

The 4096 bit array for ROM developed by Autonetics requires an area of only 64 x 96 mils. When the necessary decoders and drivers are added, the total chip area is a relatively moderate 130 x 150 mils. The ROM is organized as a 512 word by 8 bit memory. It is designed to require about 60 mw when operating at 1 Mhz. Standby power is virtually zero. Even when a memory system employing this 4096 bit ROM is active, the byte organization requires only three ROM's to be dissipating power at a time for a 24-bit word memory. A one-bit-per-word chip organization would require 24 ROM's active at a time for a 24-bit word memory.

The read access time for the 4096 ROM is less than 500 nsec with a total memory cycle time of about 800 nsec. MOS ROM speeds are a strong function of the array size, larger arrays being correspondingly slower.

2.3.1.5 MNOS Memory - Semiconductor memories have been criticized for their volatility in the case of Read/Write memories and the difficulty in changing programs in the case of ROM's. Considerable interest has therefore arisen in an electrically alterable read only memory produced using the MNOS process. MNOS memories are non-volatile and have very low standby power. Read access times are expected to be about 0.5 microseconds in large arrays but write times have thus far been very slow (1msec). The technology is therefore being considered only for ROM applications in normal computer systems.

The MNOS storage mechanism involves the switching of the threshold voltage of an MNOS field effect transistor. The hysteresis in this switching action make the devices suitable for memory applications. In order to evaluate the array characteristics of the devices, a 32 x 2 bit test array has been fabricated at Autonetics. Plans are underway to design a 1024 MNOS array during the coming year. The array will require an approximate 100 x 100 mil die and will include full address decoders and the circuits necessary for reading and writing the array. This technology is still developmental. It cannot be considered as producible in 1972 without significant risk.

2.3.2 Magnetic Memories

Magnetic memories considered for this application may be broken down into three categories:

1. Core
2. Plated Wire
3. Thin Film

The major categories, of course, can be broken down into sub-headings depending on system architecture but for this study a "2-1/2 D" memory organization was chosen for each case. This choice is best for the plated wire and thin film technologies but it is a debatable choice for core systems of less than the 16K word capacities.

Table 2-3 lists the major applicable data for 2K word and 16K word systems of 24 and 36 bits per word for core technology. Table 2-4 depicts this same data for plated wire systems. A separate table is not provided for thin film technology since the basic items are the same as for plated wire with the exception that the memory array would be smaller in size. This includes circuit counts, connections to the array, and power dissipation.

Each of the three technologies was evaluated on several criteria that should be of importance in achieving the performance goals of the reconfigurable G&C computer.

TABLE 2-3. CORE MEMORY SYSTEM DATA

		2K, 24 BIT MEMORY	2K, 36 BIT MEMORY	16K, 24 BIT MEMORY	16K, 36 BIT MEMORY
Array	Number of Mats Mat Size Array Dim.	4 128 x 96 4-1/2 x 4 x 1.3	4 128 x 144 4-1/2 x 5 x 1.3	32 128 x 96 4-1/2 x 4 x 3	32 128 x 144 4-1/2 x 5 x 3
Circuit Counts & Number of 8" x 5" Modules	Word Bit Sense T&C Total Modules	48 (1 Mod) 48 (1-1/2 Mods) 24 - - (1/2 Mod) 3	48 (1 Mod) 72 (2-1/4 Mods) 36 - - (1/2 Mod) 3-3/4	96 (2 Mods) 96 (3 Mods) 24 - - (1/2 Mod) 5-1/2	96 (2 Mods) 144 (4-1/2 Mods) 36 - - (1/2 Mod) 7
P O W E R	Logic Word Bit/Sense Total	10 15 120 145	15 15 180 205	10 15 180 205	15 15 270 295

TABLE 2-4. PLATED WIRE SYSTEM DATA

		2K, 24 BIT MEMORY	2K, 36 BIT MEMORY	16K, 24 BIT MEMORY	16K, 36 BIT MEMORY
Array	Number of Mats Mat Size Array Dim.	2 256 x 216 7.8 x 6.2 x .8	2 256 x 312 7.8 x 8.3 x .8	4 512 x 432 14.2 x 10.5 x 1.05	4 512 x 624 14.2 x 14.8 x 1.05
Circuit Counts & Number Number of 8" x 5" Modules	Word Bit Sense MLTPLX T&C Total Modules	32 (1/2 Mod) 24 (1-1/2 Mod) 24 - 24 - - (1/2 Mod) 2-1/2	32 (1/2 Mod) 36 (2 Mod) 36 - 36 - - (1/2 Mod) 3	64 (1 Mod) 24 (1-1/2 Mod) 24 - 24 - - (1/2 Mod) 3	64 (1 Mod) 36 (2 Mod) 36 - 36 - - (1/2 Mod) 3-1/2
System Volume		126 in^3	157 in^3	262 in^3	343 in^3
Array Connections		1K	1.4K	3.6K	5.4K
P O W E R	Logic Word Bit/Sense Total for 3:1 R/W Ratio	10 15 20 30	15 15 28 37	10 15 20 30	15 15 28 37

2.3.2.1 Reliability - A major item affecting reliability of a memory system is the number of connections in the memory element array. The plated wire or thin film technologies have an advantage of about 2-1/2 to 1 over a core system in this area due primarily to the number of memory elements that can be put on one plane. It was judged that core arrays no bigger than 128 x 144 could be used while plated wire or thin films could have 256 x 576 elements on a plane.

Another important item is electronic circuit complexity. Timing and control and sense amplifier circuits are about equivalent in all three technologies. Word circuits for a core system are increased by 50 percent and bit circuits by 100% to 300% over that required for plated wire or thin film. In addition, the semiconductors used for bit drivers in a core system must drive a much larger amount of current (by an order of magnitude) than the other technologies which result in larger devices or in devices being driven to a higher power level and hence, less reliable.

Perhaps a better way to compare circuit complexity than circuit counts would be to compare circuit board areas. This comparison shows about 50 percent less circuit boards are required for a plated wire or thin film memory for the larger memory capacities. This advantage decreases to 16 percent for the smaller memories. The major reason for this advantage of the plated wire and thin film memories is that more MSI/LSI can be used with them than with a core system due to the difference in bit current levels. A bit current of 40ma can be driven through MOS multiplexer switches and bipolar MSI transistors, but 400ma levels required by core systems make LSI/MSI interfaces with the memory stack impractical.

2.3.2.2 Transient Failure Immunity - The NDRO capability of plated wire or thin film memories would minimize transient failures. The only guard that would need be put in a NDRO memory would be to make sure that a write cycle would continue to completion once it had started, thus insuring a memory word is not left in some undetermined magnetic state if power transients were experienced. A core memory, being DRO, would need to be protected such that the data is rewritten after readout if transients were experienced. Due to the power level required for a core system, a weight penalty would be paid in order to protect core memory contents from loss during transients.

2.3.2.3 Cost - It appears that core and plated wire memory systems will be equivalent in the larger capacities (16K words) and at speeds of 1 μs cycle times. Price-per-bit in either case should be about five-cents per-bit. Thin film memories would be roughly 2-1/2 times this cost. Smaller sized memories would be more expensive in all cases but the increase would probably be less with a core system than the other technologies since a 3-D coincident current organization could be used

2.3.2.3 (continued) which would be more economical in the smaller capacity than a 2-1/2 D arrangement.

2.3.2.4 Growth Potential - Although none of the magnetic technologies lend themselves to true modularity, growth potential can be provided in the design. For instance, a core system could be designed so that the capacity could be increased by adding additional planes to the stack and perhaps more components to an electronics module to increase the capacity. An alternate would be to have growth potential components built into the electronic circuit modules from the start so that only an increase in array planes would be required. In any event, if growth is desired the magnetic memory design should be optimized to include the growth from the start rather than trying to make "add-ons" after the fact.

2.3.2.5 Volume and Weight - The volumes of both core and plated wire memories would be roughly equivalent for a 16K word memory capacity. A thin film unit would be on the order of 30 percent smaller. These estimates are based on space required for conventional multilayer type circuit modules in all cases. The plated wire array is assumed to be similar to that designed for present day avionics systems. The core stack is based on design numbers from core manufacturers. The thin film array is assumed to be about half the size of the plated wire array and to use the same circuitry as the plated wire system. Relative weights for a first order estimate would be proportional to the volumes.

2.3.2.6 Power - Plated wire and thin film memories have a large power advantage over a core system, requiring approximately 37 watts as compared to 295 for a 16K 36-bit system. This is due to two reasons:

1. Plated wire and thin films being NDRO require no restore cycle following readout but in a core system data must be restored after each information retrieval.

2. The digit drive is approximately 10:1 higher for a core system (400 ma vs. 40 ma). Since this current is required for each bit of the data word during a write (or restore) cycle, it is quite apparent why the core system dissipates so much more power. Word currents are about the same in each case.

2.3.2.7 Technology Criticality -

1. Core Memory Technology - This memory technology is old and very well understood. In 1972, core systems will be about like today and will be able to meet the 1 μs speed requirement using a 2-1/2 D organization in a military environment. Therefore, the technical risk of using this technology is almost non-existent.

2.3.2.7 (continued)

2. Plated Wire Technology - Plated wire is a production item at present. Examples of this in military systems are the IDCU memory (Advanced Minuteman contract) and Poseidon memory. By 1972, several good sources for military systems should be available and the technical risk low.

3. Thin Film Technology --This technology is still "just around the corner" for most companies. Although many companies have done a lot of work on it and it has even been used in actual hardware, yield problems and low output signals still are not completely solved. For a system requiring very low power and very small size, this technology might be a good choice. However, there would be a development effort and a high technical risk.

2.3.2.8 Summary - Based upon the evaluation both plated wire and core memories could meet system requirements at low or no development risk. Although similar in some areas, plated wire memories have definite advantages in areas such as power, reliability, and transient failure tolerance. Therefore, it is recommended as the prime candidate for magnetic memory systems in the reconfigurable G&C Computer system.

2.4 MULTICHIP PACKAGING OF UNCASED DEVICES

Ever since electronic equipment designers became proficient in the use of semiconductor devices, they have been fascinated by the possibilities offered by the uncased device. They have compared the size of the chip with that of the packaged device and have visualized orders of magnitude reductions in the size of their equipments. Further, they have dutifully divided wafer costs by the number of die per wafer (suitably weighted by a yield estimate) and have decided that a significant cost is involved in packaging devices. Finally, they have counted the number of wire bonds required to electrically connect the chip to the pin-outs and have foretold large increases in reliability. Thus the multiple lures of reduced costs, reduced size, and improved reliability have been responsible for the generation of a wide variety of approaches to the utilization of uncased semiconductor devices.

Autonetics has been actively developing technology for multichip packaging of uncased devices over the last several years. Various approaches to many facets of this new technology are presently being investigated and applied to uncased MOS/LSI devices developed by Autonetics. One of the key factors being developed is that of device bonding. Device bonding refers both to die bonding--mechanically fastening of the device to the substrate, and to lead bonding -- electrically connecting the die to the circuit. With discrete devices, die bonding is usually an alloying procedure which provides a very good thermal path from the chip to the substrate but increases rework cost and time and precludes salvage of the chip for re-use or for post-mortem procedures. Organic plastics have been used for this purpose in various

2.4 (continued) multichip packaging approaches. This enhances the rework procedure somewhat but generally involves relatively long curing cycles and destructive device removal. Current thinking is to eliminate die bonding and to rely on the lead bonds for mechanical strength. This is practical from the mechanical point of view; however, a penalty is incurred in terms of increased thermal resistance to the substrate. For the power levels being considered (<1 watt/chip) this appears tolerable.

Although lead bonding has generally been done with "flying lead" wires, this approach is not compatible with elimination of die bonding. In addition it is usually considered to be expensive, subject to operator error in complex assemblies, and a historical source of unreliability. For these reasons this method of lead bonding is not considered for the approach to device bonding.

Thus the three device bonding methods considered to be available in a practical sense are:

1. Thermocompression, beam lead configuration;
2. Ultrasonic, flip chip configuration;
3. Solder reflow, flip chip configuration.

Of these three approaches the first and third, namely thermocompression beam lead and the solder reflow flip chip, appear to offer the most promise. Methods of packaging uncased devices can range from conventional looking flat packs on multilayer boards to stacks of ceramic substrates interconnected with side rails.

The reliability over wire bonded devices can be partially realized even if only one device is mounted per package. In this case one of the three bonds normally required per interface lead is eliminated. Some reliability is therefore gained but the individual packages must still be soldered or welded one at a time to a multilayer board. The potential size and weight advantages of beam lead devices are entirely lost in this one device per package configuration.

Another approach is to bond a few devices on a relatively small ceramic interconnecting substrate. These are then mounted in a protective package or can and the interface connections on the substrate are bonded to the leads on the can. The cans can then be mounted on printed circuit boards in the conventional manner. The printed circuit boards can usually be high reliability two sided instead of multilayer because of the reduced interconnects made possible through this packaging technique. Many of today's advanced avionics circuits utilize a very similar packaging method. By placing several devices on a common substrate the number of bonds required to connect interface lines between devices is reduced from the conventional 6 to 2. The number of connections from the substrate to the can must be added to this. By proper partitioning of the system, however, can leads can be minimized. Substrates suitable for packaging in this manner are usually limited to a little over 1 square inch. Up to 30 conventional

2.4 (continued) bipolar IC's or 12 MOS LSI devices can be interconnected on such a substrate. Substrate interface leads can be limited to 40.

By making the ceramic substrate an integral part of the package, larger substrates with more devices can be utilized. A typical substrate of this type may be 2 x 2 inches and contain up to 40 devices. If the substrate is to be packaged singly some protective cover needs to be placed over the mounted devices with the substrate acting as the back of the package. The seal does not need to be hermetic since the devices can be nitride passivated. A portion of the edge of the substrate is left uncovered to provide connection to the outside world. This area can have fingers for mating with an edge connector. In this case the substrate becomes a miniature printed circuit board. If a large number of interface connections are required, however, more than one edge will be required and edge connection becomes awkward. An alternate is to place pins around the periphery of the substrate so that it can be plugged into a two sided printed circuit board. Perhaps four such substrates could be interconnected on one board.

To illustrate the kind of densities which can be achieved on large ceramic substrates with beam lead or flip chip devices, two substrates developed by Autonetics are described below. The first is for a 24-bit parallel Central Processing Unit (CPU) using flip chip devices. This CPU is mechanized with 23 LSIC's, some with more than 1000 Field Effect Transistors per LSIC. The CPU was laid out and interconnected on a one inch by two inch single side substrate in only two layers. The second layer metallization is for the approximately 9000 crossovers. This layout has been fabricated first using KMER as an insulation layer to evaluate alignment and tooling problems, and secondly, using silicon oxide to evaluate a practical insulation. The two mil lines and four mil center-to-center line spacing in this substrate is desirable from the function apportionment and MOS circuits; however, it is pressing the state-of-the-art for low cost interconnection.

A read/write memory board has been laid out which will contain 32 Read/Write MOS memory chips and 2 address decoder chips. The 2 inch by 2 inch board size has 0.005 inch interconnection circuit lines on 0.010 inch centers. The ceramic boards are metallized for chip interconnections. Insulation dots and crossover conductors are used to achieve specific patterns. The technology was developed by using a crossover test pattern of 0.005 inch wide conductors (chromium, gold) with insulation (silicon oxide) and 0.005 inch wide crossover conductors. Vacuum deposition was used for the first conductor layer and insulation dots. The crossover bridges were iron plated; this was necessary for continuity of the bridges over the thick silicon oxide. All of these materials were deposited, covered with a photo-resist, pattern exposed, and etched.

2.4 (continued)

Instead of packaging substrates singly, very compact systems can be constructed by stacking substrates. This concept is particularly applicable to all semiconductor computers where size and weight are critical. In systems using magnetic memories, however, the advantages are less apparent. The following paragraph illustrates this stacking concept as applied to a 16-bit parallel computer.

The advanced package contains uncased MOS LSI circuits mounted on ceramic printed circuit boards with beam leads and assembled into a computer package. This packaging design was compared in detail with conventional packaging methods (single 42 lead IC's packaged on multilayer boards) and showed the following advantages: It required one-hundredth the volume; one-tenth the weight; one-half the number of thermocompression bond joints. An advanced package test model demonstrated a temperature rise, between the heat sink and the component, that was one-fourth the estimated thermal rise of the conventional package. These savings are reflected in the improvement of the computer reliability figure (17,481 hr MTBF as against 9016 hr MTBF for the conventional package). This is the result of the reduction of bonding and solder joints and the lower temperature at which the IC's are operating.

3.0 SUMMARY OF SYSTEM ANALYSIS AND TRADE-OFFS

This section covers a summary of the requirements analysis performed in determining the guidance, navigation and control computer requirements for the space station and the trade-offs conducted in order to determine the hierarchy of the computational elements within the system. The details of the analysis and results are presented in Appendix 3.

3.1 BACKGROUND AND GROUND RULES

The study dealt with the Guidance, Navigation and Control subsystems of the Space Station. A functional diagram of the overall system is presented in Figure 3-1. Earlier analysis has indicated the desirability to dedicate a computer system exclusively for guidance and control functions. This G&C computer is the central data processor for the G&C subsystems containing various sensors and actuators required to perform navigation and attitude control of the system. Data processing functions associated with experiments and display and control functions are handled by another computer complex called the Information Management Data Processor. The latter provides mode control signals and receives navigation data from the G&C computer.

A common multiplexed data bus provides a means of transferring data between the subsystems and the guidance and control computer complex. The data bus offers not only reduction in weight, volume and power, but allows system flexibility permitting modifications and additions, standardized interfaces and increased reliability.

The organization of the computer system offers several variations between two extremes:

(1) A highly centralized system with a powerful computer complex which communicates directly with sensor elements and actuators.

(2) A highly decentralized system in which each sensor and actuator subsystem has its own local processor performing the control computations and checkout functions associated with that subsystem.

Between these extremes, there is room for several variations. For example, the local processor could be nothing more than a data compression and multiplexer device necessary for interfacing with the data bus. On the other hand, it could be a general purpose computer consisting of a processor, memory and "standard interface unit". The advantages of such a system is that although it may require more hardware, it could prove to be more cost effective because of the high degree of hardware standardization. The detailed mechanization of the subsystem functions could be

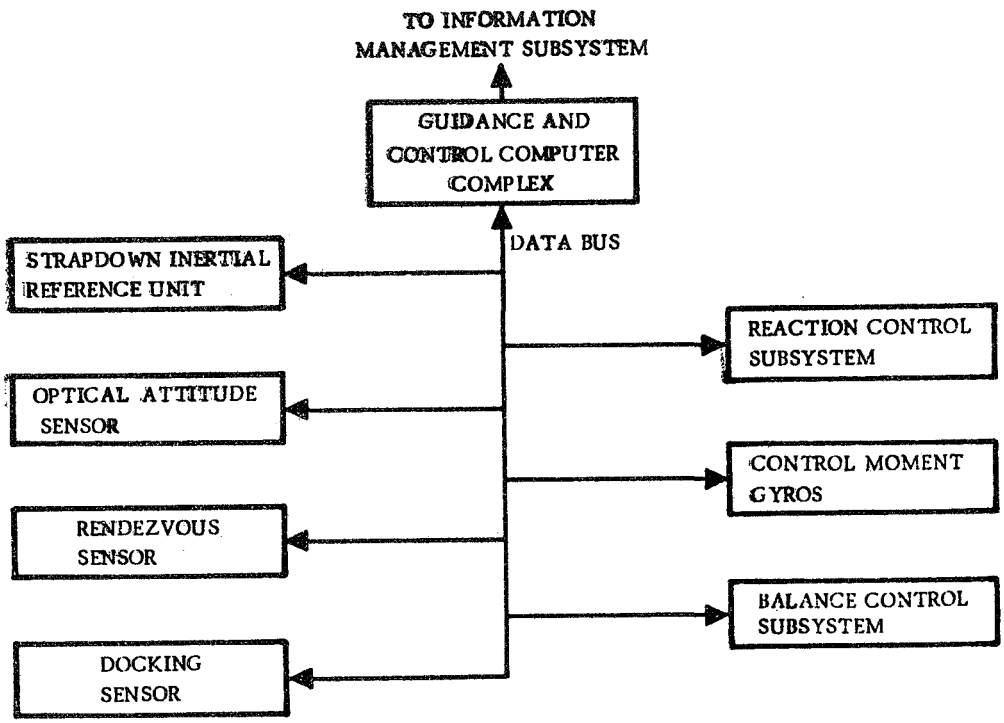


FIGURE 3-1. GUIDANCE AND CONTROL SUBSYSTEM

3.1 (continued) handled by software, permitting ease and flexibility of making changes without impacting the central data processor functions. This approach has been made more attractive by recent advances in digital computer technology which permit a substantial reduction of size, weight and power through Large Scale Integration (LSI) in logic and memory implementation.

The purpose of this portion of the study was to investigate the total computational requirements of the G&C system and determine the degree of decentralization. Four subsystems were selected for detailed analysis: (a) Strapdown Inertial Reference Unit (SIRU); (b) Optical Attitude Subsystem (OAS); (c) Control Moment Gyros (CMG's) and (d) Reaction Control Subsystem (RCS).

The trade-offs were conducted keeping the following objectives in mind:

Reduction of development risk and cost by staying within projected 1972 technology.

Reduction of management and technical interfaces between subsystems.

Reduction of development costs by utilization of standardized hardware.

Reduction of data rates on the I/O data bus.

Reduction of cost of design changes through built-in flexibility.

3.2 BASELINE SYSTEM DESCRIPTION

The baseline system as shown in Figure 3-1 consists of the following sensors and actuation subsystems:

3.2.1 Strapdown Inertial Reference Unit (SIRU)

The inertial reference system uses six single degree of freedom gyroscopes and six linear accelerometers in a dodecahedron array. The instruments are of pulse-rebalance type. The principal merits of this configuration is that it offers failure isolation of up to two out of six of both types of instruments and continuous system operation with up to three out of six failures. Furthermore, when all instruments are operating, the redundancy permits cancellation of some error sources associated with the strapdown operation.

3.2.2 Optical Attitude Sensors (OAS)

The OAS subsystem includes both star trackers and horizon scanners. The star tracker measurements are used to provide attitude corrections while horizon scanner measurements are combined with computed state vector to update the estimate of the space vehicle state. Two two-degree of freedom gimbal systems are used to hold two star tracker heads and two horizon scanner heads with each head mounted rigidly with respect to others.

3.2.3 Control Moment Gyros (CMG's)

The CMG subsystem was assumed to have three two-degree of freedom control moment gyros configured for zero net angular momentum at gimbal nulls for purposes of accommodating local vertical and artificial "g" mission modes. The subsystem is used for attitude hold and low rate maneuvering. When the gyro gimbal output axes have precessed away from the nominal valve, momentum dumping or desaturation is implemented with the RCS subsystem to restore the gimbal output axes to the vicinity of their original position.

3.2.4 Reaction Control Subsystem (RCS)

The Reaction Control Subsystem contains sixteen (16) reaction jets arranged in four orthogonal quad stations. The jets are arranged to produce pure couples about the three control axes under normal operation. The RCS is used to remove high rate transients, provide higher attitude maneuver rates, desaturate the CMG's and provide translation for orbital makeup/stationkeeping. A dual bi-propellant source is available to each RCS station and the distribution is controlled by quad valves in each propellant line. Each jet is assumed to be locally controlled by quad valves (series-parallel) in each of the fuel and oxidizer lines.

3.2.5 Rendezvous Sensor

The Rendezvous Sensor is used to provide on-board generated data with respect to distant shuttle vehicles. The data generated include range and two line-of-sight angles to the shuttle vehicle.

3.2.6 Docking Sensor

The Docking Sensor subsystem generates the necessary data for performing control during the final phases of docking. Four sets of docking sensors are shared among various docking ports. They provide range and two line-of-sight angles to the shuttle vehicle.

3.2.7 The Balance Control Subsystem

The Balance Control Subsystem will compensate for large shifts of mass on the Space Station such as shuttle vehicle docking and undocking or elevator or cargo motion. It is required only in the artificial "g" mode.

3.3 MISSION DESCRIPTION

The Space Station is expected to operate in a circular 200 - 300 mile orbit of 55 degree inclination with added capability for polar orbits. The mission can be broken down into four major phases: Prelaunch, Boost, Orbit Injection and Orbital Coast. The on-board G&C system, after participating in prelaunch checkout, will remain passive during launch until the orbit has been established. All G&C functions are being controlled by the booster during this phase. Immediately prior to transfer of control, the Space Station G&C system will require activation and checkout. After control is transferred, the G&C system will perform functions of attitude control and navigation in an unmanned orbital coast condition. References for attitude control may consist of the initial reference upon control transfer (inertial or local level) as well as other inertial or local level references to be executed upon subsequent command. The navigation function is to perform orbit determination in a primary role and to receive ground track update as an incidental role. The duration of unmanned operation is expected to be less than two days. Upon a command from an approaching Logistics Vehicle, the Space Station will hold the commanded attitude preparatory to docking and transition.

After manned entry to the Space Station, an interval of familiarization and checkout will require the G&C system to perform attitude control and navigation during orbital coast, similar to unmanned operation but with an additional provision for manual inputs.

After the familiarization interval, the booster undergoes end-to-end transposition under manual control. Next, the booster and Space Station combination is deployed and spun-up with the G&C system's only requirement during spin-up being to provide and maintain commanded spin rate. The combination is spun for artificial "g" assessment during the first month of manned operation. During this time the G&C system is to provide balance control for wobble damping, maintain commanded spin rate (appriximately 4 RPM), and correct for spin axis precession within prescribed limits. The G&C system is not required to perform navigation or state vector determination of Experiment Modules during the artificial "g" period. After the combination is despun and retracted, a zero "g" configuration under manned operation will commence.

3.3 (continued)

The zero "g" operation will consist of orbital coast with the G&C system performing functions of attitude control and navigation. Attitude reference may include local level (earth), inertial, and solar inertial. In addition, the G&C system will perform functions of state vector determinations of co-orbiting vehicles, calculations of transfer impulses pursuant to rendezvous or dispatch, steering commands to incoming vehicles during rendezvous, and translation (steering) and attitude commands to docking vehicles. Also, the G&C system will compute and issue stationkeeping commands to the reaction jets. Since the force levels will be relatively small and stationkeeping may be considered as continuous operations (being inhibited only by on-board experiments, convenience of system momentum budgets, or convenience of orbital angle) it is considered as a task to be performed during orbital coast rather than defined as a separate mode.

The orbital coast phase is the phase of primary concern for this study and is used to estimate the basic memory size, speed, and signal interface requirements for the computer system. Furthermore, there is no differentiation between the unmanned mode and the manned mode for this phase since the unmanned mode is considered a subset of the manned mode.

3.4 FUNCTIONAL REQUIREMENTS

The functional requirements imposed on the G&C computer system are shown in the top flow diagram for the overall system mechanization - Figure 3-2. To determine the computer requirements, detailed flow diagrams at further levels of detail were developed to determine the relationship between each computational subtask. Figure 3-3 presents an example of the first level of detail for the attitude determination function. For functions involving the SIRU, OAS, RCS and CMG's, mechanization equations were developed for each major block of the flow diagram and estimates were made of the number and types of instructions necessary to solve each equation. For the remaining functions, data from previous computer programs were extrapolated or new equations were derived in cases where no previous data existed. The results were compiled in terms of computer memory requirements (instructions, constants, and data) and the number of times each instruction was executed per second. The latter figure was normalized to equivalent short instruction rates, which corresponds to a number of short instructions, such as Add, per second. A long instruction, such as Multiply, was assumed to require twice the time for execution as compared to a short instruction.

A brief description of the computational functions and the method of estimating computer requirements are described below:

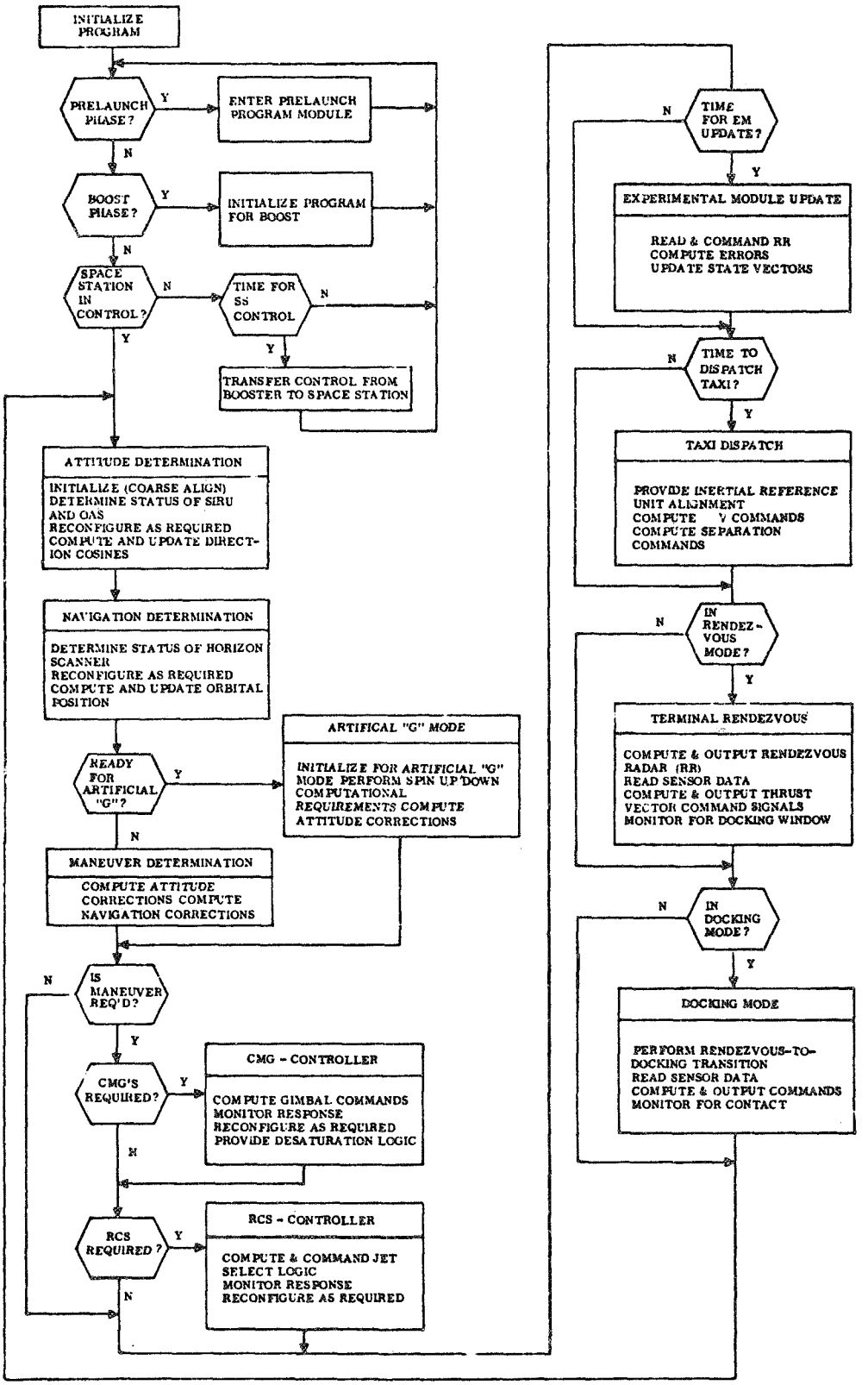


FIGURE 3-2. G&C TOP FLOW DIAGRAM

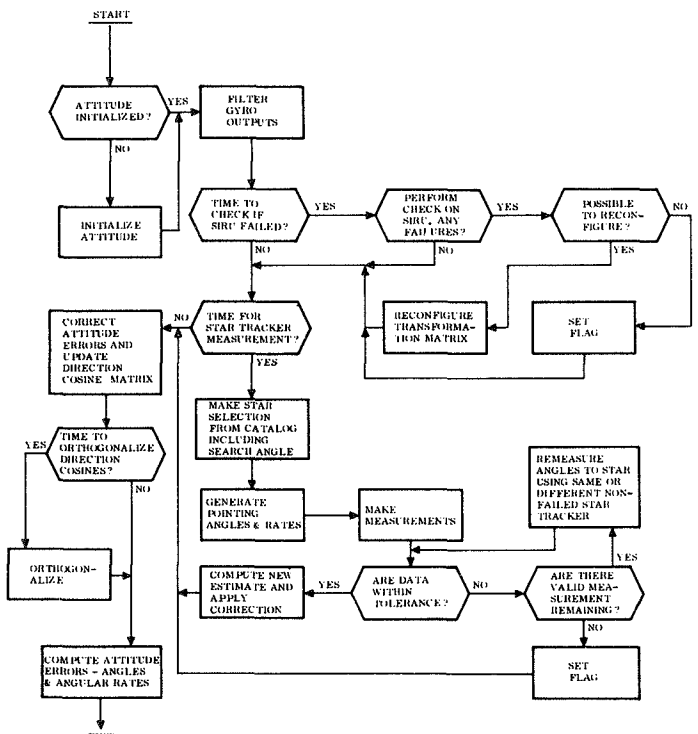


FIGURE 3-3. ATTITUDE DETERMINATION FLOW DIAGRAM - FIRST LEVEL

3.4.1 Attitude Determination

The prime purpose of this function is to provide the direction cosines for attitude control of the space station/logistics vehicle. For this study, control was provided in both inertial and local level. The two subsystems used in performing this function are the SIRU (Strapdown Inertial Reference Unit) and OAS (Optical Attitude Sensor). A first level flow diagram for attitude determination is shown in Figure 3-3. This diagram presents an example of the first level of detail in determining the computational requirements. The computer functions can be broken down into the following major computational blocks.

a. Gyro filter equations
b. Failure detection and isolation equations
c. Star selection routine
d. Star pointing command and control
e. Star tracker failure detection
f. Direction cosine update equations
g. Direction cosine orthogonalization
h. Star tracker measurement update equations

3.4.2 Navigation Determination

The purpose of this function is to estimate and update the space station position and velocity relative to the reference system. Again, the SIRU and OAS subsystems are used exclusively in computing these data. It includes the following functions:

a. Accelerometer filter equations
b. Failure detection and isolation equations
c. Delta velocity update
d. Position and velocity update
e. Integration routines
f. Polynomial prediction coefficients
g. Horizon scanner command and control
h. Horizon sensor scanning angles
i. Measurement angle computations
j. State update measurement equations

3.4.3 Maneuver Determination

This function generates the CMG's and/or RCS steering command signals for attitude and/or navigation corrections. The control law assumed for this study uses proportional plus rate and is based on the phase plane relation to minimize limit cycling. This function provides for various steering modes including:

3.4.3 (continued)

a. Hold attitude (fine or coarse)
b. Low rate maneuver (employing CMG's)
c. High rate maneuver (employing RCS)
d. CMG desaturation maneuver
e. Manual/automatic outer-loop commands

3.4.4 Artificial "G" Mode

The purpose of this function is to perform both the static and dynamic computational requirements necessary for balance control. Balance control may be viewed in two parts as static balance and dynamic balance. Static balance, insofar as practical, should be viewed as a pre-spin-up activity. However, if the "g" forces during spin-up are conducive to static balance transfer, then static balance may include the initial time period of spin. Static balance requirements could be viewed as a non-G&C system responsibility since a different system may contain the status of housekeeping layout and the extent of consumables. However, in the sense that minimizing static unbalance will minimize dynamic balance requirements, there is some justification for the G&C computer to compute static balance requirements. The computer requirements are estimated based on a model utilizing the following assumptions:

a. Spin rate control, spin-up deployment (such as cable length control if required), and spin-down retraction control are not considered a part of the balance system.

b. The CMG's will be used for wobble damping and other cyclic effects.

c. The RCS will be used for long-term drift effects such as spin-axis precessing and/or for high attitude rates.

d. A second order compensation will be considered adequate for the dynamic conditions with associated time lags between sensor response to torque generation as well as geometric displacement due to spin. Although the effects will not be comparable in all three axes, the computation requirement may be treated similarly.

e. Five spin rate conditions will be assumed in keeping with artificial "g" assessment at different levels. This assumption will correspond to five sets of constants for compensation.

3.4.5 CMG Steering

The function CMG Steering is mechanized using the H-vector control law to generate the appropriate CMG torque and momentum errors for the attitude control of the space station, and to provide for desaturation of the gyros. This function is broken down into the following subfunctions:

a. Control mode actuation logic
b. Torque error computations
c. Momentum error computations
d. Desaturation sensitivity logic
e. Failure detection and isolation
f. Reconfiguration model and logic

3.4.6 RCS Steering

The purpose of the RCS Steering function is to provide the necessary logic and computations to compute the torque and force commands, and the engine valve control, and provide failure detection, isolation and reconfiguration of the reaction control system. It is made up of the following subfunctions:

a. Control mode actuation logic
b. Torque and/or Force computations
c. Engine value control logic
d. Failure detection
e. Failure isolation
f. Reconfiguration

3.4.7 Experiment Module Update

The purpose of this function is to provide the necessary logic and computations to update the state vectors of the experiment modules (2 modules plus 1 taxi). The exact mechanization for this function is outside the scope of this study. However, for purpose of estimating, the space station's state vector computation in combination with update measurement equations for the rendezvous radar was used.

3.4.8 Module Dispatch

The purpose of this function is to provide the capability to align a simple inertial reference on board the taxi vehicle and provide appropriate commands to transport the experiment module to and from the space station via the taxi. A gross estimation was made using alignment procedure data from previous studies combined with simplified command and control equations for a co-orbiting vehicle.

3.4.9 Terminal Rendezvous

The purpose of this function is to compute the rendezvous radar look angles, process the return angles, and compute the command and control signals necessary to position the external vehicle (shuttle/taxi) in a pre-docking stationkeeping window. Extrapolation of Apollo information and simplified equations were used for computer requirements estimate.

3.4.10 Docking

The Docking function is mechanized to execute the necessary logic and computations for performing the transition from rendezvous to docking (and vice versa), and establish appropriate monitoring to provide six degree-of-freedom command and control necessary to docking the external vehicle. Limited data is available on automatic docking. Apollo docking has been manual. Some work has been performed on the AAP (Apollo Applications Program) towards automatic docking, however, no work has reportedly been done relative to AAP computer sizing and the information is not readily available. Therefore, in order to establish representative computer requirements, automatic docking equations were generated based on a 6-degree of freedom automatic docking model.

3.4.11 Computer Housekeeping

Computer housekeeping is defined as including the following functions:

a. Program Executive
b. Computer Diagnostics
c. Utility Routines
d. Input/Output Storage and Control

The estimates provided for each of these functions are based on previously mechanized programs of comparable complexity and magnitude (e.g., F-111 Avionics System).

The executive, as estimated, is structured to provide such functions as power-up power-down sequence, real-time clock control, job scheduling, transient control, etc. An estimate of 1200 words is allocated for this function.

The estimate for performing computer diagnostics is set at 1200 words. This estimate is considered sufficient to cover normal memory, CPU, and I/O type diagnostics.

3.4.11 (continued)
The estimate for the utility package (1200 words) is slightly higher than the normal avionics package due to an increase in additional utility functions.

The I/O estimate is based on the number of signals requiring storage not covered by the operational estimates. A typical example is the status monitoring words associated with each of the various subsystems and associated instructions for alerting the executive program. Also included are the command instructions for handling the data bus traffic. The estimate is based on previous I/O mechanizations which vary as a function of computer organization. The estimate given (900 words) is considered reasonable.

3.5 COMPUTER REQUIREMENTS

The results of the central G&C computer requirements analysis are presented in Tables 3-1 and 3-2. Table 3-1 represents computer requirements for a system without local processing capability at the subsystem level while Table 3-2 represents the other extreme, where the central processor functions have been minimized by performing as many functions at the subsystem level as possible. For the second case, only four subsystems were selected as candidates for performing computations at subsystem level and were subject to detailed analysis and trade-offs: SIRU, OAS, RCS and CMG's. Therefore, the computer requirements were grouped into two categories: one category of computer functions which could be performed at the central computer or in the subsystem itself, which were the subject for the detailed computational allocation trade-offs between the subsystems and the central computer complex, and the second category of computations which were not subject to a trade-off analysis.

The first category involves the RCS, GMC's, SIRU, and OAS subsystems and includes the following major functions:

a. Attitude Determination
b. Navigation Determination
c. Maneuver Determination
d. CMG Control
e. RCS Control

The remaining functions, which deal with the outer-loop command and control requirements, are categorized as follows:

a. Experimental Space Module Updates
b. Taxi (co-orbiting shuttle) Alignment
c. Terminal Rendezvous
d. Docking
e. Balance Control

3.5 (continued)

The computer requirements for the functions performed in the first category are 17,400 words of memory and 814,800 equivalent short operations per second.

The requirements for the category two functions are 21,200 words of memory and 79,200 short operations per second. The requirements listed as background in Table 3-1 are computations scheduled at intervals much greater than once per second (e.g., once every 1000 seconds). A design allowance for performing these functions and for non-periodic functions such as rendezvous and docking is estimated for background operations. Typical background computations include, for example, star selection, star pointing, star tracker failure detection, direction cosine orthogonalization and attitude update. The twenty percent duty cycle allowance is also intended to accommodate reconfiguration functions which are exercised only in case of a computer failure.

The estimated computational requirements for the case having maximum preprocessing at the subsystem level are given in Table 3-2. In this case, only those functions dealing explicitly with the SIRU, OAS, CMG's and RCS are examined, with the carry-over of requirements from Table 3-1 for the remaining functions. It must be noted that in providing this estimate, little consideration is given here with respect to the computer size and/or speed necessary at the subsystem level to arrive at these values. Such considerations are presented in the following section. The estimates provided in Table 3-2 are 7,100 words of memory storage and 263,800 short operations per second. This means that by preprocessing at subsystem level for the four specific subsystems investigated, the central computer requirements can be reduced by 10,300 words of memory and 604,000 equivalent short operations. The significant reduction in requirements is the effective reduction of speed to the level where it is well within the state-of-the art of aerospace computer technology. The reduction in memory capacity at the central computer will become more significant when one considers the fact that the central computer complex will be mechanized with redundant computers, while at the subsystem level the degree of computer redundancy might be lower than at the central processor.

3.6 COMPUTATIONAL ALLOCATION TRADE-OFFS

The objective of the computational allocation trade-offs was to determine the best allocation of computations between the central G&C computer system and local processors dedicated to the following subsystems: SIRU, OAS, RCS and CMG's. The local processor appears to perform certain functions better than the central processor. For example, such items as signal formatting, data reduction, self-test and performance monitoring can reduce the complexity of this central computer, reduce I/O bus data rates, and offer better subsystem isolation such that subsystem changes

TABLE 3-1

ESTIMATED COMPUTER REQUIREMENTS
MINIMUM PREPROCESSING

Program Module	Storage Requirements	Iteration Rate No./Sec	Equivalent Short Operations Per Second $x10^{-3}$
1. Attitude Determination	3,400	100	320
2. Navigation Determination	4,100	100	170
3. Maneuver Determination	1,100	20/200	60.8
4. CMG Control	3,100	20	64
5. RCS Control	5,700	10/200	200
Subtotal	17,400		814.8
6. Exp. Module Update	4,000		
7. Taxi Module Align	1,000	20	
8. Rendezvous	3,000	1	
9. Docking	2,200	20	41.8
10. Balance Control	6,500	20	
11. Executive	1,200		13.2
12. Diagnostics	1,200		12.2
13. Utility Routines	1,200		-
14. I/O Control	900		12
Subtotal 2	21,200		79.2
Subtotal 1	17,400		814.8
15. Background	- *		120
Total	38,600		1014.0

* Storage requirement included with other (1-14) tabulation.

TABLE 3-2

ESTIMATED COMPUTER REQUIREMENTS
MAXIMUM PREPROCESSING

	Program Module	Storage Requirements	Iteration Rate No./Sec	Equivalent Short Operations Per Second $x10^{-3}$
1.	Attitude Determination	2,100	100	130
2.	Navigation Determination	2,900	100	40
3.	Maneuver Determination	1,100	20/200	60.8
4.	CMG Control	400	20	8
5.	RCS Control	600	10/200	25
	Subtotal 1	7,100		263.8
6-14.	Subtotal 2	21,200		79.2
15.	Background	-		67
	Total	28,300		410.0

3.6 (continued) will result in no or minimum changes in the central computer software. In the previous section, the computational tasks which could be subject to allocation trade-offs were identified and grouped into category 1. This trade-off phase was concerned with the problem of determining the best split of these tasks between the central processor and the local processor dedicated to each of the four subsystems.

The following objectives and criteria were established for performing these trade-offs.

1. Minimization of hardware complexity.
2. Minimization of I/O data rates.
3. Minimization of management interface affecting:
   (a) Number of interface signals
   (b) Technical data exchange
4. Minimization of central computer load
5. Maximum reliability
6. Maximization of programming efficiency including impact of program changes.

The computational blocks which could be located either in the local processor or the central computer were listed and were grouped in such a way that they were consistent with the basic objectives of the study. The requirements for the local processor were then determined in terms of memory (both read-only memory for program storage and constants, and read-write memory for data and variables), speed (equivalent short operations per second), I/O bus data rate (number of 16 bit words/sec.), and number of interface signals (data control commands, discretes, etc.). The computational allocation was then selected that satisfied best the evaluation criteria.

3.6.1 CMG's and RCS Trade-Offs

The computational requirements estimate for the CMG's was conducted in accordance with the H-vector control law. Failure detection and reconfiguration was based on comparing measurement data of 30 signals against the response of a simulated model subjected to the same input error signals. The technique was based on the capability to reset the model on desaturation of the CMG's, respectively, thus always having a base reference to reset the open-loop model. The RCS configuration consisted of four (4) engine stations having four (4) bi-directional engines per station. Failure and isolation was implemented by monitoring transducers (pressure and temperature) strategically located in the dual redundant bi-propellant fuel lines, across the quad-redundant control valves, and on the engines themselves. Both subsystems, as previously mentioned, have their associated computation requirements divided

3.6.1 (continued) into six major computational program modules as listed:

	CMG's	RCS
A	Control Mode Detection	Control Mode Detection
B	Torque Error Computation	Torque/Force Computation
C	Momentum Error Computation	Engine Valve Control
D	Desaturation (Momentum Dump)	Failure Detection
E	Failure Detection & Isolation	Failure Isolation
F	Reconfiguration	Reconfiguration

Ten (10) different combinations of these modules from maximum to minimum were evaluated for both subsystems respectively. Of the ten different cases, six (6) each are selected for discussion here.

3.6.1.1 CMG's - The amount of preprocessing estimated for the CMG's, Figure 3-4, does not impose any real stringent requirement if all of the processing were performed at either the subsystem level or in the central processor. However, in viewing the criteria of minimizing the load on the central computer and minimizing management interface, processing at the local level is recommended. In allocating the most optimum split and in keeping with the trade-off criteria (minimum I/O data rates and number of data signals) the suggested split is the allocation configuration given for case 3, where:

Allocation Configuration	Computational Program Modules Performed at Subsystem Level
1	None
2	A, B, C, D, E, F
3	C, D, E, F
4	D, E, F
5	E, F
6	E

3.6.1.2 RCS - The allocation configurations evaluated for the RCS are given in Figure 3-5. However, the requirements estimated and presented in this figure represent an LP (Local Processor) configuration having triple redundancy and servicing all four stations. In this configuration, the requirements are considered to be a maximum and relatively stringent on the local processor. However, based on the same attributes given for the CMG's, case three is recommended as the optimum split, where:

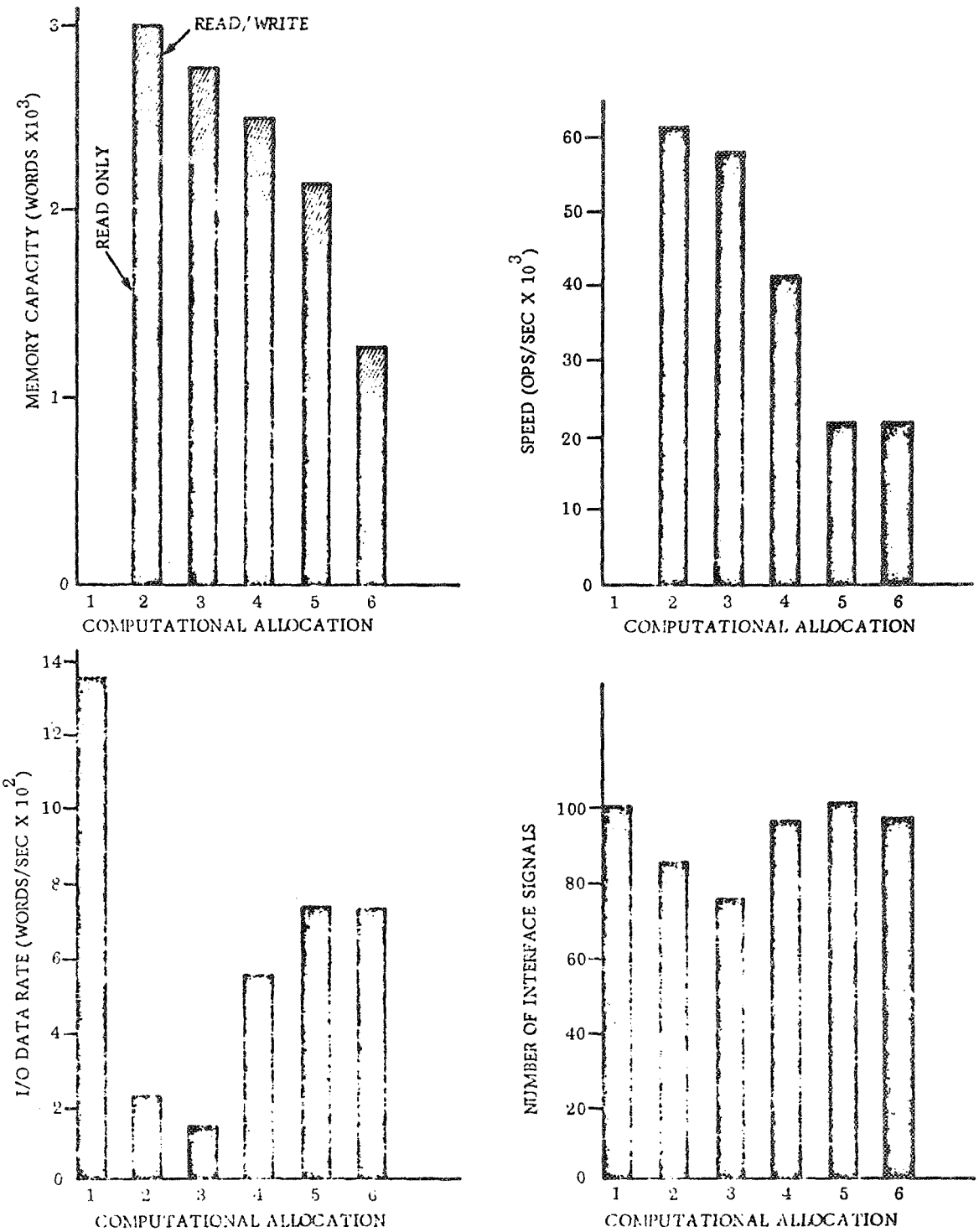


FIGURE 3-4. CMG COMPUTATIONAL REQUIREMENTS VERSUS PROGRAM ALLOCATION

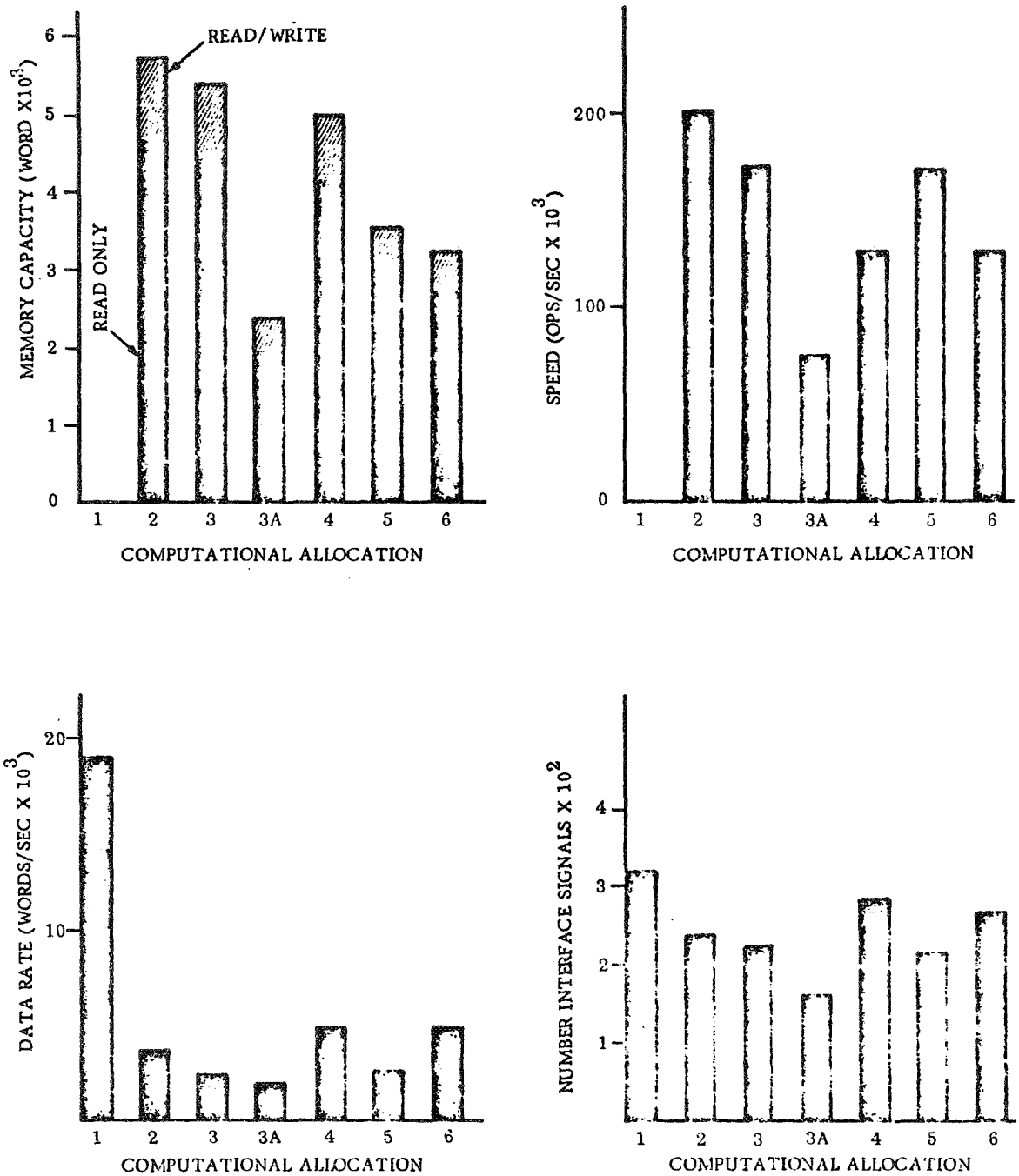


FIGURE 3-5. RCS COMPUTATIONAL REQUIREMENTS VERSUS PROGRAM ALLOCATION

3.6.1.2 (continued)

Allocation Configuration	Computational Modules Performed at Subsystem Level
1	None
2	A, B, C, D, E, F
3	C, D, E, F
4	D, E, F
5	C, D, E
6	D, E

A second LP configuration (3A) became very attractive and was evaluated later in the study. This configuration consisted of dual LP's located at each engine station. In this configuration, the requirements imposed on the LP were reduced significantly in the area of memory and speed, especially (See Figure 3-5). The reduction is attributed to the distribution of failure detection and reconfiguration requirements over four (4) separate LP configurations. This configuration is much more desirable in that it reduces the LP requirements to be within those specified for the CMG's (commonality), and from the aspect of having the computers located near each engine station. That is, the engine stations are separated by many feet and would need long leads or a sophisticated sub-multiplexing system for data transfer between station electronics and the centrally located LP complex.

The recommended configuration and allocation split is still in conformance with case 3 above. This allocation offers the same attributes previously discussed with an even greater magnitude. A larger number of the data signals (152) estimated, represent failure and reconfiguration flags (discrete signals) and are anticipated as a requirement for on-board checkout recording where processing is performed at the subsystem level. The data rates and data signals are somewhat higher when considering four LP locations as opposed to one.

3.6.2 SIRU and OAS Trade-Offs

The computational requirements estimated for the SIRU and OAS, center around performing the inner-loop attitude control functions and the outer-loop guidance/navigation functions. For the candidate systems specified, the following computation modules were selected as appropriate break points in the various computations for allocation trade-offs.

3.6.2 (continued)

SIRU

A. Filter Instrument Outputs
B. Failure Detection and Transformation to Body Coordinates
C. Direction Cosine Matrix Update
D. Direction Cosine Orthogonalization
E. Generation of Attitude Error Signals

OAS

A. Failure Detection
B. Compute Horizon Sensor Scanning Angles
C. Process Measured Data
D. Compute Horizon Sensor Pointing Angles and Rates
E. Compute Star Tracker Pointing Angles and Rates
F. Make Star Selection

A basic ground rule in making these allocations was to assign to the central computer those computations that are independent of data from the subsystems and/or computations that involve one or more sensors. And in keeping with this rule, the following computations are explicitly assigned to the central computer:

1. Attitude Update Using Star Tracker Data
2. Integrated Position and Velocity
3. Position and Velocity Update from Horizon Measurement
4. Maneuver Determination

The trade-offs for both subsystems involved sequentially cascading each of the modules into the LP and accumulating the memory and speed requirements and defining the interface for each module.

3.6.2.1 SIRU - The requirements with respect to minimum preprocessing at the subsystem level, as shown in Figure 3-6, involves only the data rate and data signal requirements necessary to perform the computations in the central computer. In this configuration, it is recommended, however, to accumulate the foregoing pulses at the subsystem level for an obvious reduction in the data rates. That is, the 10 kc accelerometer pulses should be accumulated and transmitted at the update rate commensurate with the direction cosines (specified at 100 times/sec. per this study).

For the case where maximum processing is performed at the subsystem level, configuration 6, the speed requirement is approaching the state-of-the-art. The major contributing factor for the excessive speed requirement is the 100 times per second update rate assumed for this study. An analysis, although very limited, indicates that an update rate of ten (10) times per second in the case of the space station environment would be more than adequate for both the inner and outer-loop control.

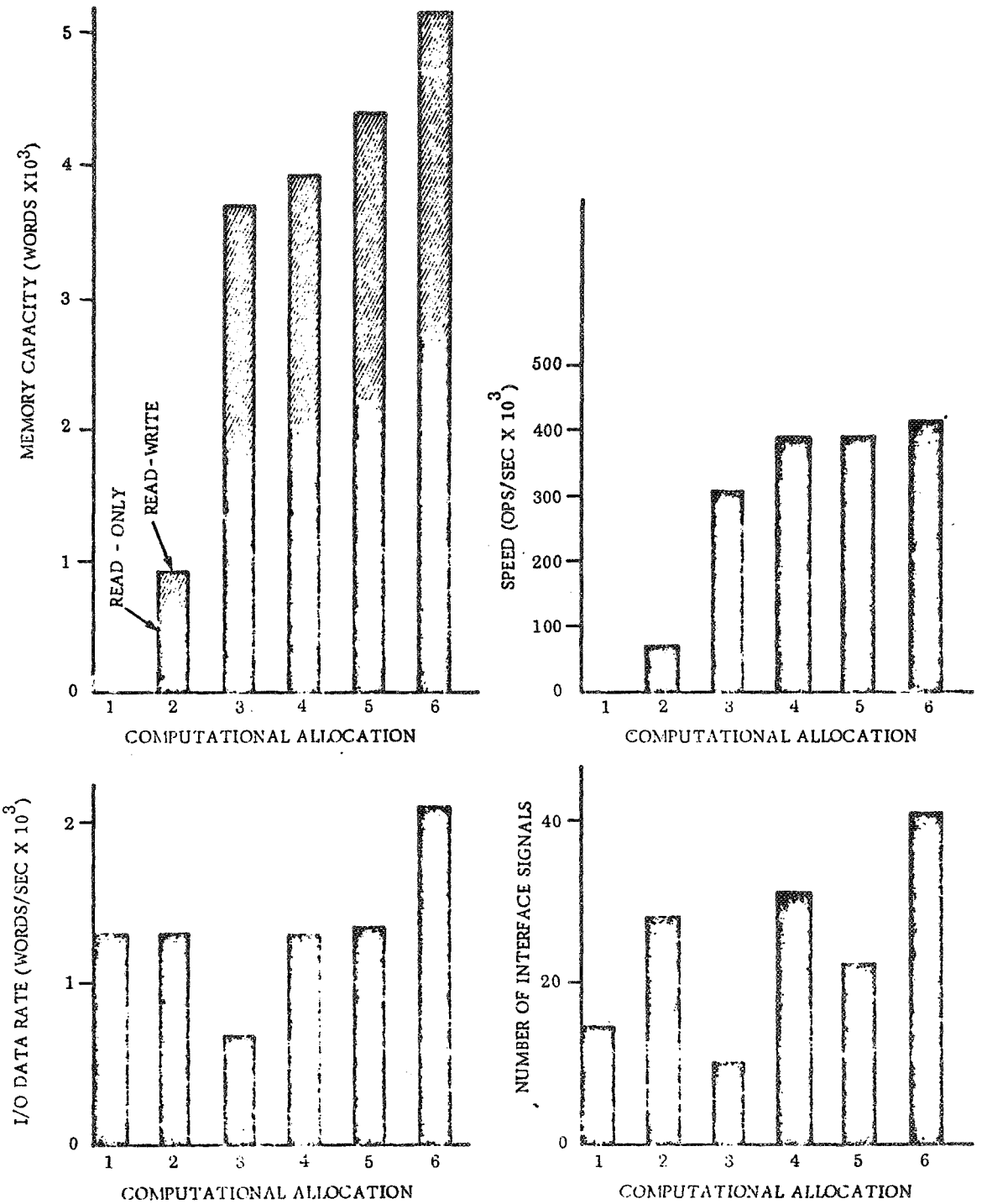


FIGURE 3-6. SIRU COMPUTATIONAL REQUIREMENTS VERSUS PROGRAM ALLOCATION

3.6.2.1 (continued) For this reason, along with the above mentioned attributes, the recommended computational allocation is configuration 5 which includes all but one of the five program modules given, where:

Allocation Configuration	Computational Program Modules Performed at Subsystem Level
1	None
2	A
3	A, B
4	A, B, C
5	A, B, C, D
6	A, B, C, D, E

Generation of the attitude error signals, program module E, is recommended as being performed in the central computer. The argument here is that the error signal outputs are required for CMG and RCS actuation commands and effectively come under the basic ground rule of two or more subsystem involvement. If the update rate were reduced from 100 to 10 times per second, the LP configuration recommended, Case 6, would fall into or below the same class of LP recommended for the CMG's and RCS. In any case, a minimum reduction of two to one for the update rate is recommended based on the analysis performed under this study. The attributes concerning the recommended case are typical of those given for the CMG's. That is, subcontractor isolation, ease of subsystem buy-off at subcontractor's facility, minimum total system integration problems, and process designing amenable to the subsystem redundancy.

3.6.2.2 OAS - The system mechanization employed in this study requires measurement data from both the star tracker and horizon scanner at nominally very slow rates (on the order of once every 100 seconds and greater). Consequently, recommending the use of local processing is totally based on reduction of management interface and failure detection and isolation at the subsystem level. Figure 3-7 presents the results for the cases evaluated under this study. In any event, Case 7 is suggested as an optimum split relative to unloading the central computer and reducing management interface, where:

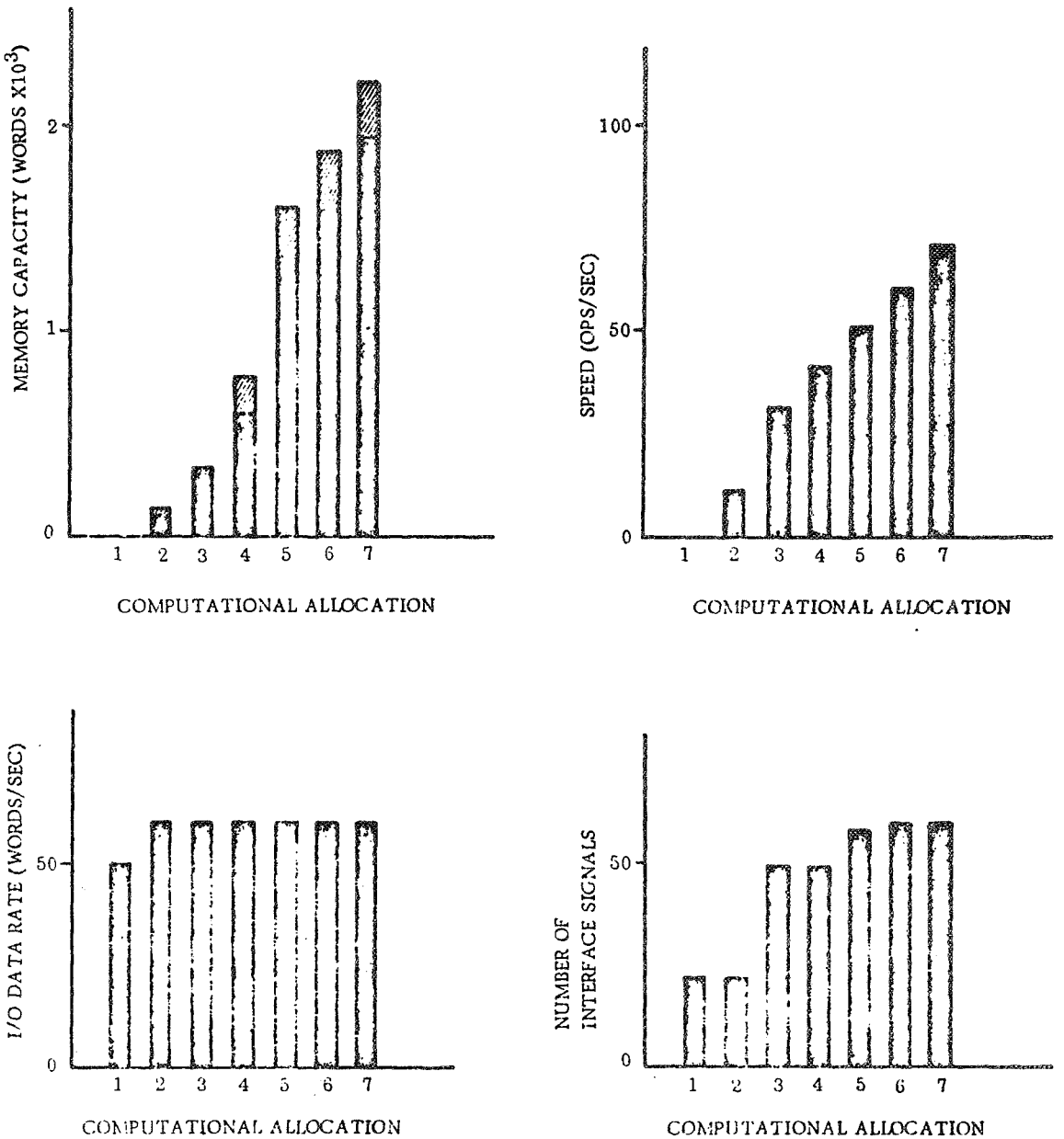


FIGURE 3-7. OAS COMPUTATIONAL REQUIREMENTS VERSUS PROGRAM ALLOCATION

3.6.2.2 (continued)

Allocation Configuration	Computational Modules Performed at Subsystem Level
1	None
2	A
3	A, B
4	A, B, C
5	A, B, C, D
6	A, B, C, D, E
7	A, B, C, D, E, F

3.7 SUMMARY AND CONCLUSIONS

The foregoing analysis indicates that the central G&C computer requirements can be reduced to a level where they are within the state-of-the-art of present aerospace computer technology by employing local processors in subsystems. By allocating the computations, as shown in Fig. 8, the central computer complex will require as a minimum 29,600 words of memory and be capable of executing 500,000 operations per second with the recommended computational allocation. The data rate requirement for the I/O data bus is reduced to approximately 88,000 bits per second. This figure represents only the actual data transmitted and does not include any overhead such as control, address and error detection and/or correction bits. A standardized local processor design with functional characteristics shown in Table 3-3 can satisfy the preprocessing requirements of the subsystems investigated without resulting in a proliferation of on-board computer systems and excessive development costs. The local processor can be mechanized with state-of-the-art LSI technology and would introduce no significant size, weight and power penalties in the overall G&C system.

The approach offers several advantages over conventional centralized computer organizations presently employed in aircraft avionics systems:

1. Reduction of Development Risk. The requirements can be met by state-of-the-art computer technology of moderate speed and memory capacity.

2. Management Interface Clarity. The sybsystem interfaces are reduced to a level where they can be explicitly defined early in the program. Subsystem checkout and sell-off is greatly simplified by this approach.

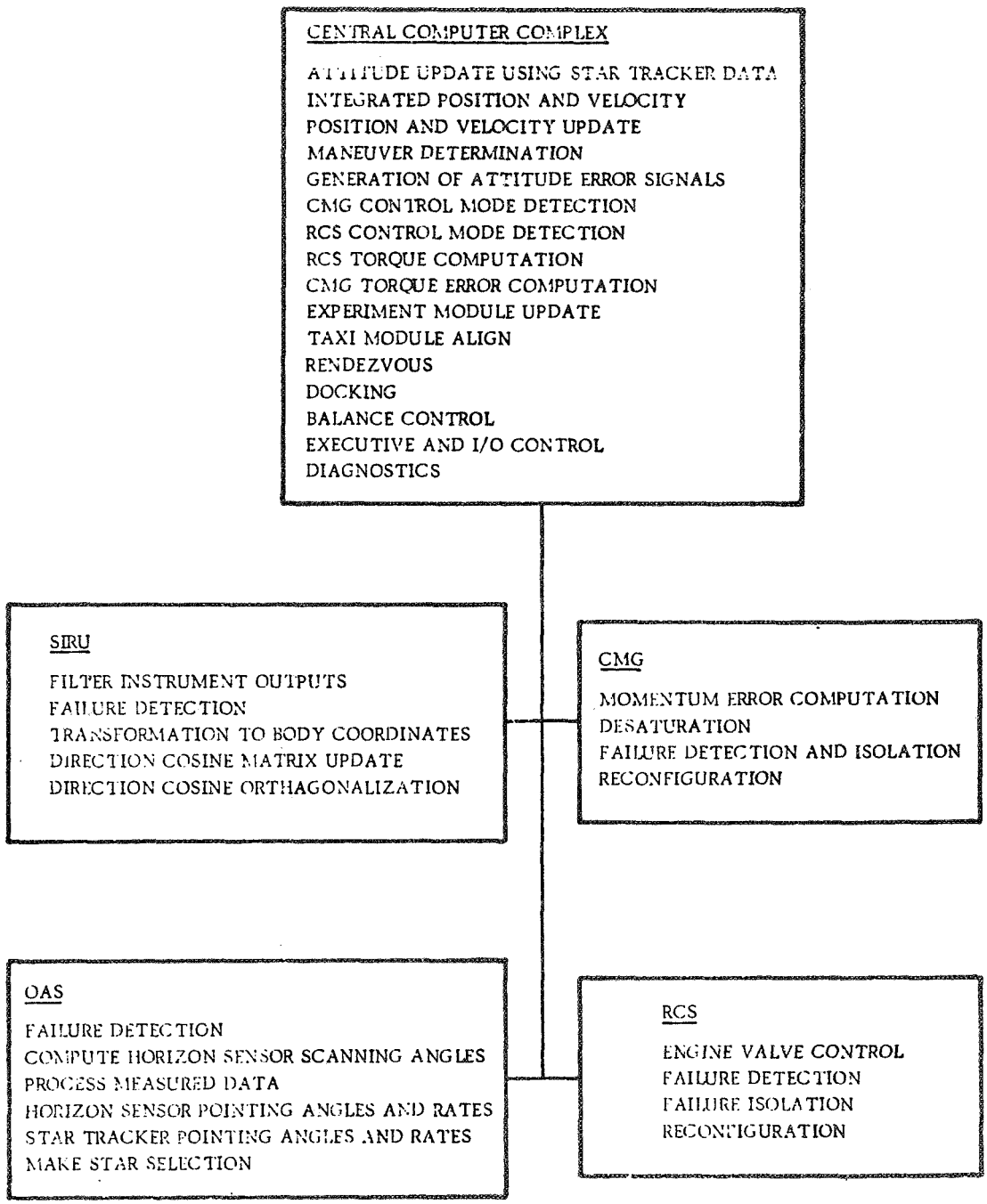


FIGURE 3-8. RECOMMENDED COMPUTATIONAL ALLOCATION

3.7 (continued)

3. Reduced Data Bus Rates. The traffic on the data bus is greatly reduced. A bus system designed for 1 MHz data rate can adequately handle any overhead and expected growth requirements.

4. Reduced Development Cost. Standard local processor design will reduce or eliminate requirements for any special purpose logic required at the subsystem level.

5. Flexibility. Most subsystem hardware changes can be absorbed by the local processor programs and will not reflect back into the central computer programs.

6. Growth Potential. The system can be expanded easily by adding more local processors.

7. Programming Ease. The subsystem supplier can program his own local processor, since he is most familiar with the computer requirements at that level and will be able to handle subsystem changes at minimum cost.

TABLE 3-3

LOCAL PROCESSOR FUNCTIONAL REQUIREMENTS

Word Length	16 bits
Memory	
(Read Only)	4096 words
(Read/Write)	512 to 1024 words
Speed	
Add Time	2.5 μsec.
Multiply Time	10 μsec.
Instruction Set	Conventional plus double precision add, subtract, store, fetch

4.0 DEFINITION OF CANDIDATE COMPUTERS

The objective of task 4 was to arrive at a set of candidate computers capable of meeting the Reconfigurable G&C Computer requirements and provide data on the set of candidates suitable for performing an evaluation to select the preferred candidate. The most important requirement placed on the Reconfigurable G&C Computer is the reliability or failure tolerance criterion, namely the fail op-fail op-fail safe (FOOS) requirements. This requires that the first two failures be tolerated so that the system remains operational and that the third failure be tolerated so that the system remains safe.

First, the basic approaches to meeting the FOOS requirement will be discussed followed by the development of computer system concepts to satisfy the FOOS requirement. Finally the definition of candidate computers along with quantitative data will be presented.

4.1 INVESTIGATION OF FAILURE TOLERANCE REQUIREMENTS

4.1.1 Introduction

This task was initiated by conducting a survey of past activity related to redundancy and reliable computer design. Most of the work to date has concentrated on the treatment of single failures or a limited number of multiple failures. (Reference 4-1 contains an excellent survey of the subject matter) Further, due to this concentration much of the activity is not applicable to modern LSI semiconductor technology. The fail op - fail op - fail safe reliability requirement imposes stringent requirements on the redundancy schemes to be considered in the study. It requires that the failure detection schemes provide 100 percent probability of failure detection and whatever switching scheme is proposed for reconfiguration be 100 percent effective; in other words, failure is tolerated.

In order to proceed with task 4 in a meaningful manner it was necessary to provide a thorough definition of the reliability requirements on the computer system. The next section presents these definitions. Investigation of approaches to meet the requirements led to the evaluation of some basic redundancy schemes; these are also presented below.

4.1.2 Reliability Requirements

4.1.2.1 Definition of Failure - The type of failures that are considered for the fail op - fail op - fail safe criterion is defined as a "module" being a single failure. A module's capability is varied in this study, spanning the spectrum from a single computer to a memory, I/O unit, etc., and to byte size sections of a memory. It is apparent when considering the nature of electronic equipment that this is the only reasonable assumption. This equipment is typically constructed from integrated circuits,

4.1.2.1 (continued) - some discrete components, some form of interconnect board (multilayer, or printed circuit), and some form of interconnect between boards. A single failure event may take place within an integrated circuit and not cause another component to fail, or on the other extreme a single failure event may take place in an interconnect board (e.g., shorting two planes) appearing as if a multitude of individual components (e.g., integrated circuits) failed.

Therefore, a single failure is treated as a worst case condition that an entire module has failed where the module typically has the capability noted above. A single failure is also defined as being independent of other single failures. That is, one module having failed will not result in the failure of another module. This is the key to defining a single failure and the module size. It is apparent that a module must be considered on a reasonable scale such as central processor unit. This also imposes special design considerations on the interface of various modules in the computer system such that the single failures are truly independent.

This definition of a single failure is highly critical to the direction of the study because of two factors: (1) the fail op - fail op - fail safe requirement implies that all failures be treated, and (2) all practical failure detection techniques assume one or some bounded number of single failures that are independent of other single failures.

Having defined what constitutes a single failure, the study may then proceed in an orderly manner to meet the reliability requirements. Failures such as a meteorite destroying the computer are not defined as a single failure if they affect more than one module. These types of failures that result in multiple failures by the definition above are not to be considered in meeting the fail op - fail op - fail safe requirement. However, as a ground rule the computer system design is based on the assumption that the computer system must be split among two compartments of the spacecraft. This affords some failure tolerance of catastrophic type events that can result in multiple failures.

4.1.2.2 Failure Tolerance - The computer system is designed to tolerate failures in a fail op - fail op - fail safe manner. Each fail is defined as a single failure for the purpose of this study. This also leads to the consideration of time between failures. It is assumed in the study that two or more single failures do not occur simultaneously and are spaced by minimum time intervals of approximately one second. This definition is intended to exclude, for design purposes, single failures that randomly may occur nearly simultaneously (milliseconds) while including single failures that may occur within several seconds and longer.

4.1.2.2 (continued)

Fail is also defined to include all possible modes of a single failure. That is 100 percent failure detection is required for a module. This is intended to include both permanent and transient failures.

Fail op is defined as operationally performing the critical computations after any failure. Two variables may be considered at this point: (1) The time allowed between the occurrence of a failure and becoming operational again, and (2) the amount of the total computations that are considered critical with regards to the fail op criterion. With respect to the first variable, it is assumed that the time to be operational after a failure occurs is somewhat less than one second, i.e., the switching time must be very short. This time is also expected to depend on the phase of the mission and the nature of the critical computations; the switching time defined above is expected to be a worst case condition. With regards to the second variable, the percentage of computations that are considered critical, it is expected to depend strongly on the phase of the mission. It is assumed that this may vary from the entire computational load to a very small percentage of the load.

Fail safe has been defined to include a confidence level for rapid reconfiguration. Fail safe requires that upon the occurrence of any failure, it be detected,and the reconfiguration be rapid for most failures (Ref. 4-2). Reconfiguration for fail safe requires that only a portion of the computational capability required for all the critical computations be properly operating after a failure. In other words, degraded modes are acceptable for fail safe operation. The exact percentage of the total requirement that may be considered fail safe has not been specified. The time allowed to be properly operating in a fail safe mode, after the occurrence of a failure, is the same as in the fail op case (typically, milliseconds) for most of the failures (nominally, this has been set at 95%). The goal is to be properly operating after all the failures; however, for the remaining 5% (approximately) that are not reconfigured rapidly, more time may be taken to reconfigure.

4.1.3 Computer Organization Considerations

4.1.3.1 Impact of Basic Requirements - The reliability requirements defined in the previous section were evaluated with regards to computer organizations. Figure 4-1 contains a chart illustrating some of the effects of the requirements on the approaches to computer organizations. The failure definition and failure tolerance requirements have been discussed above. These requirements lead to treating two cases: fail op- fail op - fail safe (FOOS) and non-critical failure tolerance (where a probability may apply).

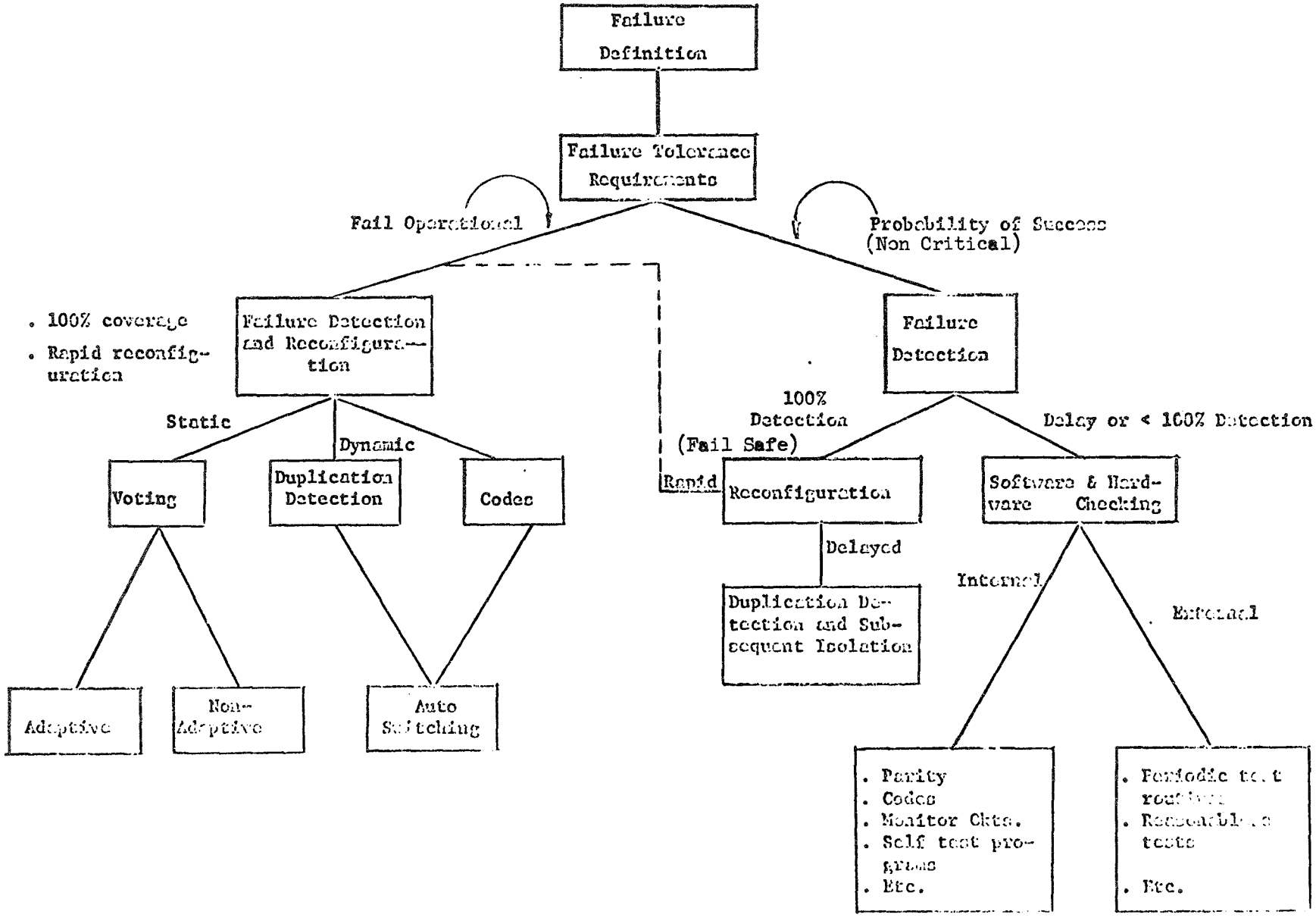


FIGURE 4-1. FAILURE DETECTION AND RECONFIGURATION APPROACHES

4.1.3.1 (continued) -

To meet the FOOS requirement the approach taken to failure detection and reconfiguration must provide a probability of failure detection of 1.0, a reconfiguration time that is less than one second, and a probability of successful reconfiguration of 1:0. For all practical purposes this requires that the failure detection be accomplished external to a module. Schemes that employ hardware redundancy and/or software self test routines internal to a module for purposes of self failure detection cannot be relied upon to provide detection for all possible failure modes of a module. (Most past studies relied on one or both of these techniques, Ref. 4-3 thru 4-9, and are therefore not applicable).

This leads to massive redundancy to accomplish detection, i.e., modules are replicated to detect failures by comparison of redundant output signals. Two schemes to accomplish this are indicated in Figure 4-1, namely, static and dynamic. The former employs voting techniques and the latter duplicate comparison for failure detection. Voting requires a majority to make a decision, ie, two out of three, three out of five, etc. It provides for detection by a majority vote with reconfiguration inherent in the voting process. Duplicate comparison techniques employ a comparison of the outputs of two modules, if they disagree, the discrepancy provides failure detection. To meet the rapid reconfiguration time, this disagreement detection is then used to automatically switch in a third module. Of course, enough modules must be provided to handle three failures as dictated by the FOOS requirement.

As indicated in Figure 4-1 codes may also be used to meet the FOOS requirement. Codes may be used for the failure detection mechanism; however, replication is required to provide a spare module for automatic switching in order to accomplish reconfiguration. The use of coding may have some merit for failure detection in modules such as memories and will be discussed in more detail later in this section.

The alternate failure tolerance requirement is for non-critical failure tolerance as shown in Figure 4-1. As discussed in the previous section, it is expected that the failure tolerance requirements will actually vary during the mission depending upon the mission phase and functions being performed, ranging from practically all fail op to practically non-critical. Failure detection for non-critical failure tolerance is dependent on two additional parameters: speed and coverage. On one extreme is the need for rapid failure detection (perhaps as fast as for the fail op case) for all failures and on the other extreme is the allowance for delayed failure detection or less than 100% failure detection with a rapid detection time. If rapid failure detection is required, then the speed for reconfiguration must also be considered. Again, reconfiguration could be rapid or delayed.

4.1.3.1 (continued) --

Rapid reconfiguration with 100% detection begins to merge into the requirements of the fail op case as indicated in Figure 4-1. Delayed reconfiguration allows duplication to be used for detection and subsequent isolation at a later time to decide which of the two modules failed.

Considering the situation where detection may be delayed or less than 100% failure detection coverage is required, opens up many possibilities to detection and reconfiguration. It should be noted here that the percentage of failure coverage is an extremely difficult topic to evaluate. While a failure detection method may provide a percentage coverage of 95%, this applies while the detection scheme is being used. If the failure is not detected by the test method, it may very well be detected at some later time as the outputs of the failed module are used as inputs to some other system. In other words as time increases, the probability of the failure being detected will approach 1.0 . There are many software and hardware techniques that may be applied to detect failures under these requirements; as indicated in Figure 4-1, parity, codes, and software test routines are just some of the methods available. Reconfiguration may also be accomplished with a combination of hardware and software techniques employing isolation, switching and reinitializing of modules (Ref. 4-3 thru 4-9 employ many of these techniques).

The intent of Figure 4-1 is to provide an overview of the approaches to computer organization, from a redundancy viewpoint, that may be taken depending on the failure tolerance, failure detection and failure reconfiguration requirements imposed on the computer system. It is expected that a mix of most of the various requirements depicted in Figure 4-1 will be imposed on the computer system under study.

The fail op requirement will be discussed below with reference to the voting, duplication, and coding methods, indicated in Figure 4-1, that may be used to satisfy this requirement. Since the fail op is the pacing requirement, fail safe will be treated as a subset of it.

4.1.3.2 Application of Basic Approaches - The voting and duplication methods are applied in Figure 4-2 for the case where a module is a single computer and one module is required for the total computational load. A non-adaptive voting organization requires five (5) modules to meet the FO FO requirement. The modules C_1 through C_5 are each assigned the same computational job and the majority voter performs a 3 out of 5 vote on the outputs. The organization can tolerate any two module failures and continue operating (in fact, at full capability after the second failure). Fail safe cannot rely on using the voter unless a total of seven (7) modules are provided.

4.1.3.2 (continued) -

The majority voter, of course, must be made redundant so that it will withstand the required number of failures and continue to operate. In fact, this redundancy should be carried to the output interface. Simply providing one output interface will in most likelihood result in a single point of failure. Preliminary evaluation indicates four or five output interfaces are required at a minimum. It should be noted here that it is possible to carry this concept further and consider the majority voting function external to the computer system. Then the computer organization reduces to a set of computers operating in parallel. It is not the intent, at this point, to discuss the voting or interface mechanization in detail but simply to point out the modular structure of the different organizations. (Section 4.2 will treat this topic in detail.

The adaptive voting organization shown in Figure 4-2 requires four (4) of the same computer modules used in the above case (1 less) The difference in this organization is that the majority voter is a two out of three voter with four (4) inputs. This requires that the voter be adaptive in the sense that it recognizes a discrepancy in one of the three inputs it is voting on and disregards that input from then on. Another module is then used for the third input to be voted on. Again, two module failures may be tolerated. At a slight increase in complexity of the voter, one of the five modules in the non-adaptive case have been eliminated.

A non-adaptive duplication organization is shown in Figure 4-2. As in the non-adaptive voting case, five of the same modules are required. The boxes labeled DD are disagreement detectors, which simply do a comparison of the outputs from a pair of modules. Two such disagreement detectors are required. A discrepancy indicates failure in one of two modules, isolation to a module is not provided; the correct outputs are obtained from a pair of modules with no disagreement detected. In the case of both pairs of modules having failed, the module C_5 provides the correct output.

This duplication concept may be extended to an adaptive case in which (4) modules are used as shown in Figure 4-2. The disagreement detector in this case detects a disagreement between a pair of modules and then uses another module to provide the correct output signals as in the non-adaptive case above. The detector subsequently isolates the failure to one of the pair of modules by comparing each module of the failed pair to the current pair of operating modules. The good module of the failed pair may then be used as a spare module to back up further failures. It should be noted the above voting and duplication schemes solve the fail op - fail op case. The fail safe requirement can be satisfied with no increase in the number of modules for the adaptive cases. For the non-adaptive cases one more module may be required to satisfy fail safe operation.

The module size in the above cases was considered to be a single computer. A module may also be mechanized at a lower level, for example; memory (M), arithmetic processor (P), and input/output processor (I/O) modules may be considered. In addition, the individual

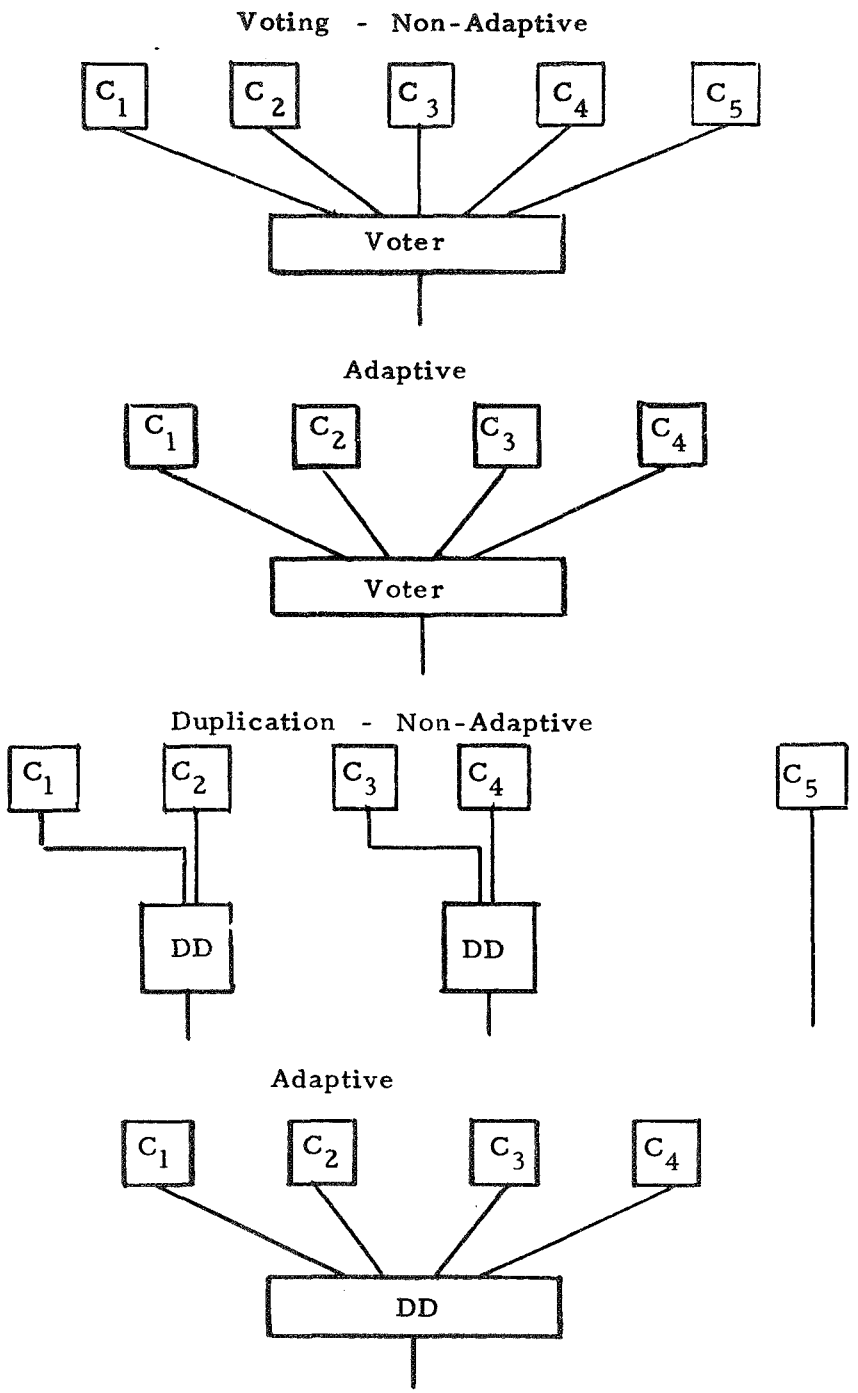


FIGURE 2. VOTING AND DUPLICATION ON COMPUTER MODULE LEVEL

4.1.3.2 (continued) - capability of these modules could be varied. In each case, at this lower level of modularity, the organization to meet the FOOS requirement will assume the same structures as given in Figure 4-2. As an example, Figure 4-3 indicates the required structure where three modules are used, M, P and I/O. This example assumes that the capability of each module is such that, one of each type is sufficient to handle the total computational load (same assumption as in Figure 4-2). It is seen that the same structure as required in Figure 4-2 is used here. As mentioned previously, the voters of course must be redundant. Many possibilities exist in the design of the voters; for example; voting on inputs or voting on outputs. It is not the intent at this time to consider the voters in detail but merely to indicate the structure of the modules to meet the FOOS requirement. (the next section discusses the problems of voting)

The above schemes may be considered to be more or less massive redundancy since they require at least duplication of a module to detect failures. There is another approach that may be considered for failure detection which is in a different direction than the massive redundancy approach. The detection scheme involves the use of checking codes, e.g., residue codes. It has application to modules such as a memory and is depicted in Figure 4-4. The memory system in this figure is divided up into a set of byte modules. Each byte represents a portion of a word from memory. In the example, four bytes represent the memory word. If the word length is 32 bits, each byte would represent an 8 bit "slice" of the memory. Each byte is also mechanized by a module. The code used for checking is also mechanized in a module. For each word stored in memory a code is also stored, therefore, all modules operate in parallel when reading or writing in the memory system. A code checker monitors the output from the memory system. The checking code can be designed to provide failure detection for any failure mode of one module. It will provide this detection with much less than a complete duplication of hardware. As indicated in Figure 4-4, spare modules (bytes) may be used to replace the failed modules. The code checker itself must be redundant so that it does not compromise the system. Further, a complete replication of this memory system must also be provided as shown in Figure 4-4 in order to accomplish reconfiguration. The memory system with code checking is self-failure detecting; once the failure is detected another replicated system must be turned to in order to provide a correctly operating system. Therefore, from a failure detection standpoint, this approach has some merits; however, reconfiguration requires that massive redundancy be resorted to such as duplication.

It must be recognized that some form of the structures presented above must be present in any of the candidate organizations in order to meet any FOOS requirements.

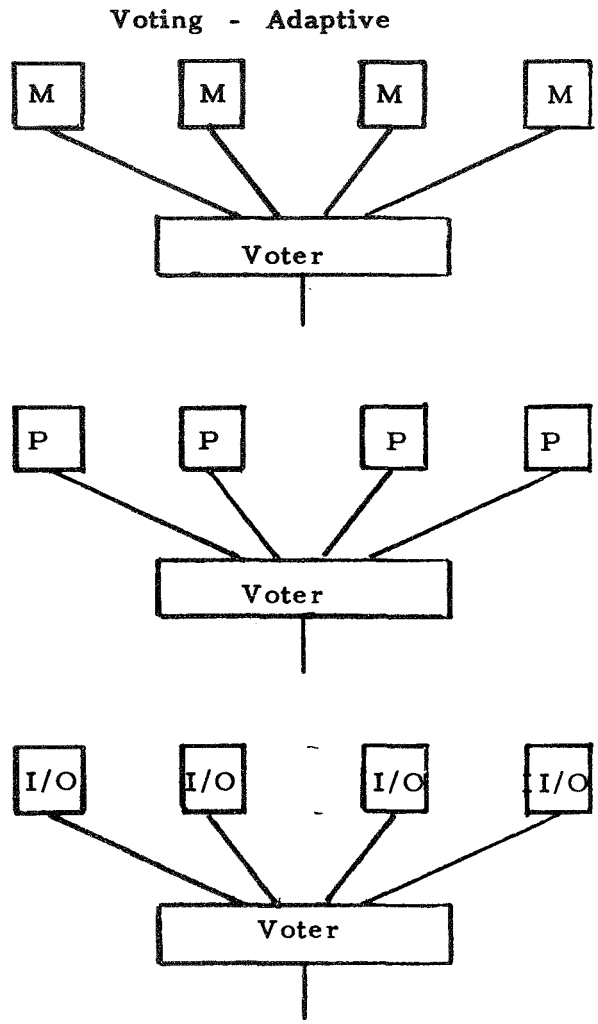


FIGURE 4-3 VOTING AT LOWER MODULE LEVEL

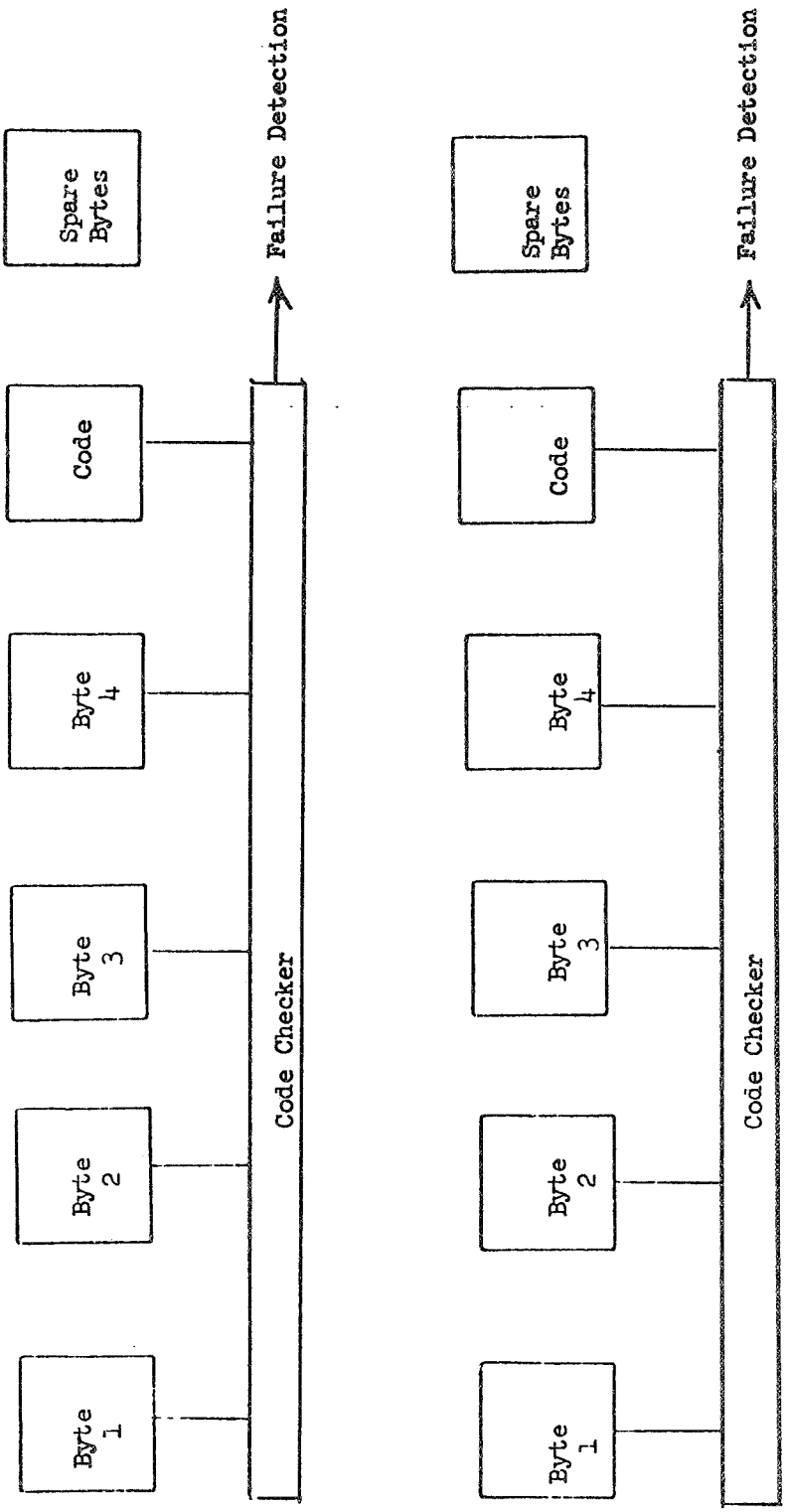


FIGURE 4.4. USE OF CODES FOR DETECTION

4.1.3.2 (continued) - From the present investigation voting and duplication appear to be the only reasonable solutions to meeting the FOOS requirement. Further, the adaptive case has the most appeal since it can meet both the fail op - fail op and the fail safe requirement with four (4) modules.

4.2 DEVELOPMENT OF COMPUTER SYSTEM CONCEPTS

4.2.1 Introduction

In the previous section it was found that a minimum level of redundancy of four using adaptive voting or disagreement detection is required to meet the FOOS requirements. Somewhere in the G&C system, the level of redundancy may be less than four. In this case, one is faced with transitioning a boundary of a higher level of redundancy to a lower level. Particular attention must be paid to such a transition in order to prevent severe over-designs or under-designs with regard to redundancy and failure tolerance. The FOOS boundary, insofar as the G&C computer system is concerned which requires a level of redundancy of four, extends up to the point of interface with the subsystems (LP's). That is, it includes the computer system and the bus system. The level of redundancy at the subsystem side of the boundary will depend on the particular mechanization of the subsystem and its functional interrelation with other subsystems.

Since the key to meeting the FOOS requirement is massive redundancy (4) with some form of a decision process (voting or disagreement detection), a good deal of attention was focused on this topic to define the computer system concepts. A straight-forward and simple attempt at a solution would be to simply provide four single computers providing four busses to the subsystems and require the subsystem to perform an adaptive majority vote. Even a simple approach such as this has problems as will be discussed below. There are many approaches as will be discussed below, primarily involving the method of interconnections and the way the decision processes (hereafter referred to as "voting") are implemented.

4.2.2 Definition of System Concepts

4.2.2.1 Introduction - Three computer/bus configurations and three bus/LP configurations have been defined as representing a large class of possibilities in design of the system concepts, these form a set of nine basic system concepts. It is not the intent to necessarily restrict the final system configuration to one of the nine candidates. The candidates have been chosen simply to provide an organized approach to evaluation of the system.

4.2.2.2 Ground Rules - The following ground rules were established for the definition and evaluation of the candidate system concepts:

1. The computer system will consist of four (4) separate units and will interface with four (4) data busses. Further bus and/or computer module redundancy will be considered independently.

2. All candidate systems will be designed to interface with subsystems whose redundancy requirements dictate from one (1) to four (4) LP's.

3. Subsystem failures on a function basis shall not cause computer system failures (real or apparent) which would be counted in the three failure survivability criteria imposed on the computer system.

4. The central computers in the candidate systems will be pictured as self-contained, independent units. However, in subsequent sections, module level reconfiguration capability will be considered.

5. Candidates will be described only up to the LP interface. LP to subsystem interconnection will be treated separately.

6. Computational requirements for a given subsystem will vary in criticality and hence, in redundancy requirements and allowable reconfiguration time, from highly critical to non-critical. For study purposes, these categories of criticality have been identified:

   a. No interruption of output data is permitted.
   b. Interruption is permitted for periods less than 5 cycles.
   c. Interruption is permitted for extended periods.

7. The "Ten Pin Rule" will apply. As currently defined, this limits the number of interconnections between modules to 50 pins maximum. Configurations resulting in 50 to 200 interconnections may be considered but are undesirable.

8. The computer system will be physically split into two separate compartments. Communication between the compartments will be primarily on the bus and communication at memory cycle speeds would appear to be unfeasible.

9. Majority voting and/or comparison on computer outputs will be utilized for failure detection.

10. All communication on the busses is initiated by the computer system, i.e., input from a LP only occurs as a result of a request from the computer system.

4.2.2.3 System Configurations - The three computer/bus configurations are shown in Figure 4-5 and called candidates 1, 2 and 3. The three bus/LP configurations are shown in Figure 4-6 and called candidates A, B and C. The characteristics of these are summarized below:

Candidate 1 - Each of the four busses is dedicated to one of the four computing units. LP participation in voting is implied since the only data path between computers is through LP's.

Candidate 2 - Each computer can transmit on only one bus but can receive information from all busses. A level of voting is possible at the computer/bus interface without LP participation.

Candidate 3 - All computing units can both transmit and receive on all busses. Some type of switching and/or voting function is implied at each bus terminal.

Candidate A - Each LP is connected to only one bus. Any subsystem level voting required must be accomplished beyond the LP's.

Candidate B - Each LP is connected to all four busses. Some level of voting/switching in the LP is implied.

Candidate C - LP's are selectively connected to from 1 to 4 busses. Voting function at LP's would vary from subsystem to subsystem.

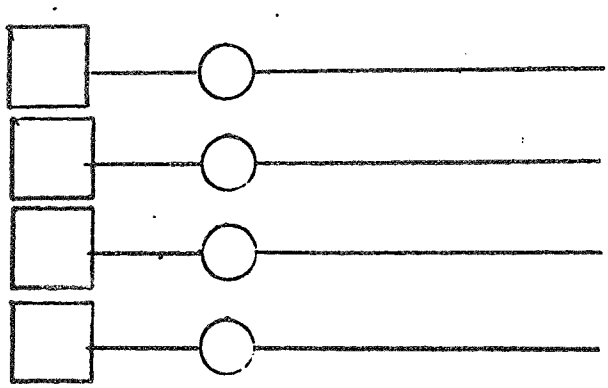
4.3 INVESTIGATION OF SYSTEM CONCEPTS

The system concepts defined above were subject to an investigation from an operational, software, and hardware standpoint. This section presents the results of this investigation.

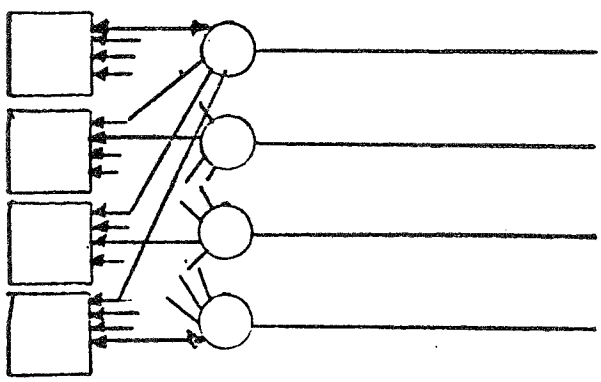
4.3.1 Assumptions

4.3.1.1 Software - For purposes of this study the anticipated requirements have been categorized in a general sense. The following definitions were applied:

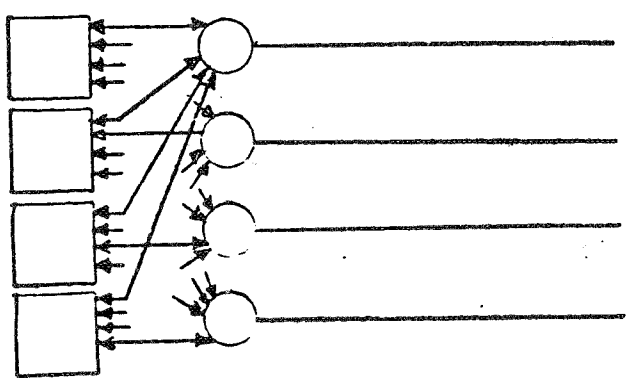
> Critical Function or Computation - The software routine(s) necessary to perform the computations for a given system function without regard to redundancy. The total of all critical functions equals the operational requirements where the term "operational" is used as in fail-operational.
>
> Non-critical Function or Computation - The software routine(s) necessary to perform a function which may be required of the computer system but which is not part of the operational requirement and hence need not survive computer failures. These include such "background" functions as may be interrupted or discontinued if sufficient computational capability is not available due to failures or an extremely high load of critical computations.



CANDIDATE 1



CANDIDATE 2



CANDIDATE 3

 = Computers.

 = Bus Terminal

FIGURE 4-5. COMPUTER/BUS CONFIGURATIONS

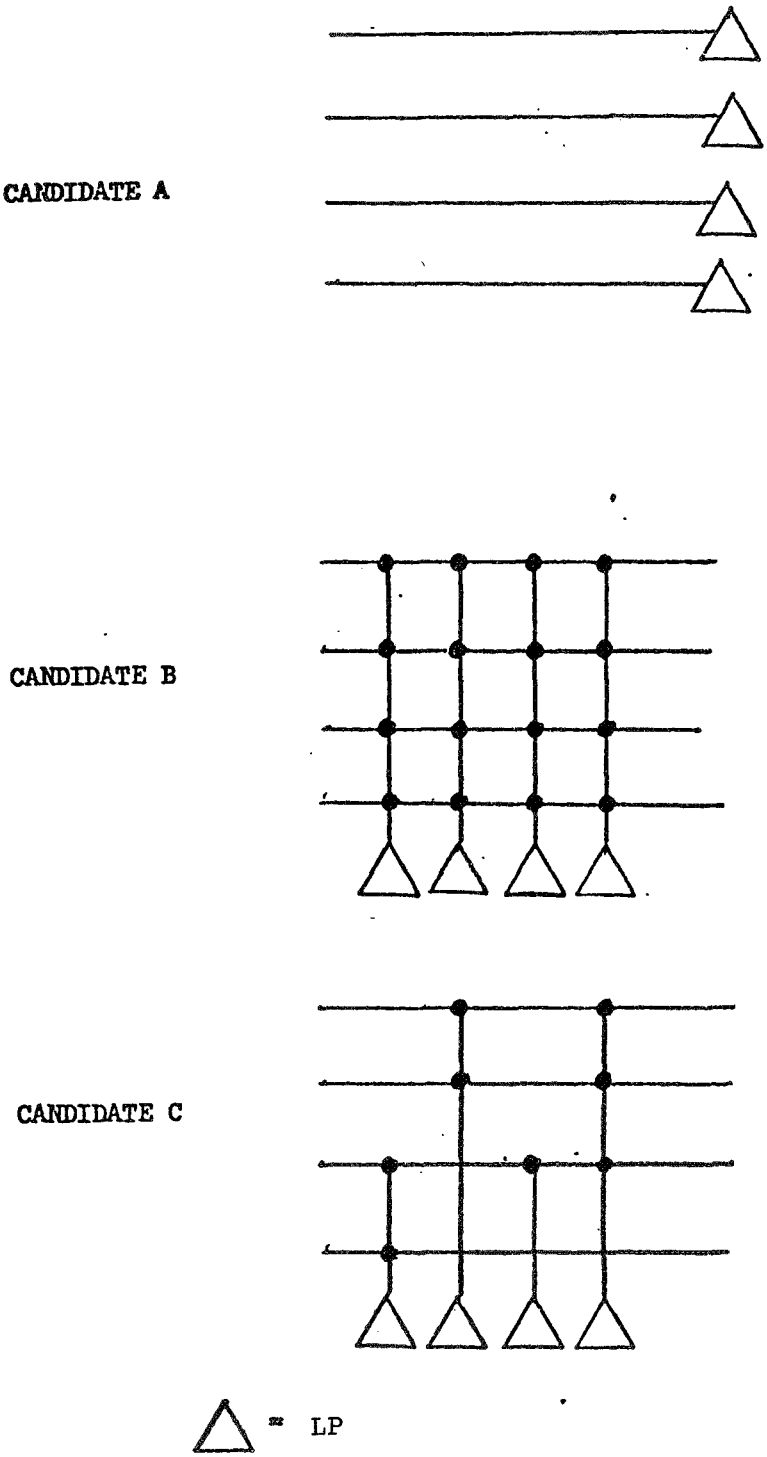


FIGURE 4.6. BUS/LP CONFIGURATIONS

4.3.1.1 (continued) -

Critical functions are further categorized according to the sensitivity of the external subsystems to errors or gaps in the output data from the function.

Three levels of error sensitivity have been defined. Note that a given function may be placed in different levels dependent on the overall system mode and/or external conditions.

> Error Sensitivity Level 1: This level allows interruption and/or errors in computational output data for extended periods of time, possibly until manual repair or intervention is accomplished. This implies that either the function is non-essential or errors in its operation can be readily detected and circumvented external to the computer system (by an operator, for instance). In the latter case, when the computer system is informed of the error, it would attempt reconfiguration. Computations which fall into this category may be singly computed and will be referred to as having a redundancy requirement of one or as R1 functions.
>
> Error Sensitivity Level 2: This level allows interruption (but not errors) in the computational output data for periods not exceeding 5 or 6 update cycles. Computations in this category must be at least doubly computed to allow output comparison and hence failure detection. Reconfiguration time for these functions must obviously be less than the 5 or 6 cycles specified. These functions have a redundancy requirement of two and will be referred to as R2 functions.
>
> Error Sensitivity Level 3: This level allows no interruption or errors in computational output data. Computations in this category must be at least triply computed in order that output voting can be performed to detect and isolate failures allowing selection of a correct data set. These functions have a redundancy requirement of three and are referred to as R3 functions.

The nature of the computations, critical and non-critical, is assumed to cover a wide range in terms of time-criticality and in terms of arithmetic/logical characteristics.

4.3.1.2 Bus/LP Operation - As previously indicated, it is assumed that the system has at least four (4) independent busses. These busses interface with Input/Output Processors (IOP's) in the computer system and with LP's at the subsystems. The Bus/LP interface must be designed to provide isolation to prevent LP failures from inducing bus failures.

4.3.1.2 (continued)

The bus system operates in a requested data fashion where all data transfer in either direction is initiated by the computer system. Some capability for data buffering and/or voting is assumed potentially possible in LP's, but reducing the requirement for these capabilities is considered desirable.

Synchronization of data on the four busses is assumed possible and in fact is the technique preferred. This synchronization may be either bit-by-bit or on a frame or time slot basis, however data on the busses may be staggered by some number of bits to preclude identical errors induced by a common error source.

4.3.1.3 LP/Subsystem Operation - All subsystem/computer communication is assumed to occur over the bus system with LP's providing the subsystem/bus interface. A subsystem may be a data source, a data recipient, or both with respect to the computer system. Moreover, the inputs from any given subsystem to the computer system may be required in order to generate the outputs to other subsystems.

Subsystems may be inherently redundant, functionally redundant, or have no redundancy whatever. LP redundancy for a given subsystem may range from one to four and is not necessarily correlated with the redundancy of the subsystem itself.

In cases where a given subsystem and/or the LP's for that subsystem are redundant and the subsystem is a data source, the data from corresponding redundant elements, though correct, may not be identical due to analog/digital conversion variation, timing differences, etc. This potential difference will subsequently be referred to as resolution uncertainty in the data.

Two alternate mechanizations have been assumed for operating a subsystem with redundant LP's.

Alternate 1: Redundant LP's are operated one at a time where the particular LP in use at a given time is controlled at least indirectly by the computer system. The implication of this mode of operation is that detection of subsystem failures can be accomplished by some means other than data comparison/voting, e.g., internal or external testing of the subsystem or modeling of predicted subsystem responses. Therefore, it will be assumed that the computer system can independently determine the accuracy of a single data set from such a subsystem. A further implication of this mode of operation is that a voting process requiring agreement of a majority of the computers will be used to control switching of LP's.

4.3.1.3 (continued) -

Alternate 2: Redundant LP's are operated in parallel and may be operating from a single or redundant subsystem. The implication of this mode of operation is that the computer system is supplied redundant input data sets which represent the same parameters but which may differ due to resolution uncertainty in the data. In general, a means of determining data accuracy other than comparison of redundant data will not be available. Intercommunication of the redundant LP's is a possibility but is not assumed to be the general case.

4.3.2 General Considerations

While investigating the configurations, many considerations were found to apply to the majority of the configurations. These considerations are presented at this time rather than with the discussion of each of the configurations.

4.3.2.1 Simultaneous Failures - One of the basic ground rules applied to the study states that simultaneous failures (separated by less than one second) of two or more computer modules will not be considered when evaluating the ability of the system to survive FOOS. A ground rule such as this seems reasonable when applied to the occurrence of failure events.

In attempting to provide maximum flexibility of the system, many critical functions may be computed triple redundantly in order to free the fourth computer for non-critical functions until such time as it is needed to replace a failed unit. While being used as a spare, the fourth computer could experience failures that would go undetected either because little redundant computation was being performed or because the failure did not exhibit itself in the computations or testing being performed. Any subsequent failure in one of the three "critical computation" computers will cause the fourth computer to pick up the failed computer's functions, but it may immediately fail because of the earlier undetected failure and will appear to the system as simultaneous failures.

Another type of failure may occur which affects only a portion of the program or is sensitive to a particular data configuration. This type of failure will be referred to as a mode sensitive failure. Mode sensitive failures may also appear as simultaneous failures although the actual failure events may have occurred hours apart.

Simultaneous failures can result in three undesirable situations with respect to a given data set.

1. The two failed units produce results which disagree with each other and with any good computer(s) performing the same function.

4.3.2.1 (continued) -

2. The two failed units produce results which disagree with each other but one of which agrees with the good computer(s) performing the function.

3. The two failed units produce identically incorrect results.

An analysis of these situations indicates several potential problems mainly related to two failed units producing identically incorrect results or results which agree with a non-failed unit. For example, take the case where a R2 function is being computed, the spare computer has an unobserved failure, and one of the two active computers fails. Now, when the failed spare is initiated to resolve the discrepancy, it will "vote out" the wrong computer in the case where failed units generate identically incorrect results.

There are many considerations associated with preventing and/or circumventing seemingly simultaneous failures. One particular consideration seems worthy of note here. Since failed units producing results which agree with other units, failed or unfailed, cause the most significant problems, it is important to consider the probability of this situation occurring. For continuously varying arithmetic functions dependent on multiple inputs, the chances of two independent failures affecting the computations in a manner which changes the results identically should be reasonably small. However, logical functions which result in one of a small number of possible conditions such as a binary, on-off decision, would demonstrate a rather high probability. One approach to reducing the probability in these cases would be to include redundant data with the results of the logical function. Computation of the redundant data would be based on a more complex function related to the input parameters for the logical function, thus providing an error code of sorts on the function.

4.3.2.2 Voting Methods - The failure detection requirements as defined imply decisions on output data. Three general methods of accomplishing voting (and hence data selection) for the decision process have been identified and are discussed below. A more detailed description of implementation techniques is presented later (Para. 4.3.2.4 and Section 4.4).

4.3.2.2.1 Voting at the Bus/LP Interface - If the FOOS boundary is extended to include the bus system, and the conclusion that 100 percent failure detection requires data voting is valid for bus failures, then performing data comparison and voting at the Bus/LP interface, i.e., within the LP, would seem to be a necessity. In the simplest case where all functions are being computed in parallel by all four computers and reconfiguration of the computer system is not required, no further action need be taken to meet the FOOS requirement. If reconfiguration is desired

4.3.2.2.1 (continued) - at the computer or computer module level, then it is necessary that the results of the vote be transmitted from the voter (LP's) to the four computers. The computers can then decide when and how to reconfigure (para. 4.3.2.5).

Since each LP is voting on the data it receives and is sending the results of that vote back to the computers, the computer system is required to resolve problems arising when the voting results disagree between LP's i.e., it must indirectly diagnose LP/bus failures.

4.3.2.2.2 Voting at the Computer/Bus Interface - Several of the candidate systems are interconnected in a manner which would permit performing a vote on the outputs of the computers to the bus system. This would be accomplished by allowing each computer to monitor outputs from the other computers so that at least three computers would vote on the validity of data on each bus. This voting would be performed simultaneously with or lagging the output of data to the bus system and consequently would not prevent transmission of incorrect data for the current transmission period. The voting results could be sent on the same busses as data or four separate vote busses could be used. If the FOOS boundary is at the bus/LP interface, then a second vote must be taken at the bus/LP interface to detect and correct bus failures. In this case, the only apparent advantage of also voting at the computer/bus interface is that bus failures are more easily identified. This second vote would also detect any incorrect data that might have been transmitted.

If the FOOS boundary is at the computer/bus interface or if techniques such as error coding can be relied on for detection of bus failures, the second vote (at the LP) would be unnecessary. However, as previously mentioned, incorrect data will be transmitted on a bus for one cycle after the failure occured. Since R3 functions cannot tolerate any loss of data, the LP must be directed to the proper bus from which to obtain data. A means of obtaining this direction, other than voting itself, is for the computer system to transmit at periodic intervals, no less frequent than the end of each data transmission block, the result of each computer's vote on the data. In this case, the LP, as a minimum, is required to buffer at least two sets of data and to input from three different busses the voting results of at least three of the computers. It must then compare the three sets of votes to determine which of the two sets of data is bad, if any. It would seem that a LP capable of performing this function (comparison of votes) could just as easily compare the data sets. If this is the case, then operating in the manner just described seems to offer only one slight advantage over the previous method and that is a potential reduction in execution time and buffer storage requirements at the LP since only three votes need to be compared rather than three entire data sets. However, it is possible that with a synchronous bus system, data could be compared automatically as it is received thus eliminating the need to buffer more than one set of data.

4.3.2.2.2 (continued) -

For R2 functions which can tolerate a single cycle or more of data interruption, the LP's would only be required to buffer one set of data and examine two sets of votes. But since comparison of the votes is still required, there is little advantage gained over voting at the bus/LP interface.

4.3.2.2.3 Computer Interface Voting with Transmission Control - Voting at the bus/LP interface appears to require a fair amount of complexity in the LP and/or voting device. Simply voting at the computer/bus interface and transmitting the results did not significantly reduce this complexity, primarily because the LP is still required to perform its own vote either on the data or on the results of the computers' votes. If this comparison of the data and/or the computers' votes can be eliminated, then a significant advantage would be gained.

It appears that the only hope of reducing voting requirements at the LP while still retaining the FOOS capability would be to design the system such that sufficient information is present on a single bus to determine the validity of the data on that bus. Any other approach implies monitoring multiple busses and performing some sort of voting in order to select the correct one.

Clearly, to achieve an independent bus approach requires that more than one computer be able to transmit on a given bus and that multiple computers have control over a transmission through a voting process. A potential mechanization of such an approach is presented later.

The approach is intended to retain the FOOS capability in the computer system while allowing a LP to receive data on a single bus with no requirement to compare data or votes. Note that such an approach does not preclude transmitting on multiple busses for some or all LP's.

4.3.2.2.4 External Voter - The actual comparison of output data that is required for voting may be performed by the computers at the computer/bus interface or by the LP at the bus/LP interface. This seems reasonable since the computers certainly are capable of doing it and in many cases the LP will also be capable of performing the vote with little or no additional hardware required.

If it is desirable an external voting device which is separate from the computers and the LP's could be used. It was determined that such an external voting device would not change the characteristics of voting at the two different interfaces. The results of the voter would have to be sent to the computer and to the LP to effect reconfiguration. Since it represents a single point of failure, four of the devices would be required and the computers and LP's would still have to vote on whether to accept the external voters' decisions. The only benefit of the external voter

4.3.2.2.4 (continued) - appears to be remoteness which can include bus failures further down the line. However, this type of voter is not very amenable to selective computational redundancy since it would have to be capable of changing voting modes for different blocks of data which would also complicate adapting. A mechanization such as suggested for the LP would seem to be required.

4.3.2.3 Input Voting

4.3.2.3.1 General Discussion - In general, multiple copies of data generated by a given subsystem will be transmitted to the computer system. The copies may be transmitted from one LP or redundant LP's and from one subsystem or redundant subsystems. The data is in many cases used by more than one computing unit in the computer system to generate redundant outputs to the same or different subsystems. These redundant outputs are at one or more points in the system compared for purposes of failure detection.

The redundant input data sets received at the computer system from a given subsystem may differ as a result of two situations: (1) an external failure (bus, LP, subsystem), or (2) resolution uncertainty in the data. In both cases if the differences are not resolved prior to use of the data in a computation, a failure will be indicated by one or more output voters. (Output voting is assumed to be based on correct data sets being identical, since modeling of the affect of allowable input variations on outputs would in general require a rather complex voting process.) If the cause of the apparent failure was resolution uncertainty in the input data, then the failure indication is erroneous. If the cause was a failure external to the computer subsystem, then it is necessary to isolate the failure so that internal computer system elements are not mistakenly discarded through reconfiguration and so that external reconfiguration can be accomplished.

The most obvious means of solving both problems is to provide capability in the computer system for voting on the redundant input data so that a single data set can be chosen and used by all computing units, thus insuring that output discrepancies represent actual failures. This requires that all input sets be supplied to all computers.

In order to determine the necessity and/or benefits of input voting, the two problem situations mentioned above were investigated to determine if solutions other than input voting are available, and are discussed in the following paragraphs. In both situations assume a configuration such as pictured in Figure 4-7 where input voting is not possible. Assume that data from Subsystem I is required to compute data for Subsystem II.

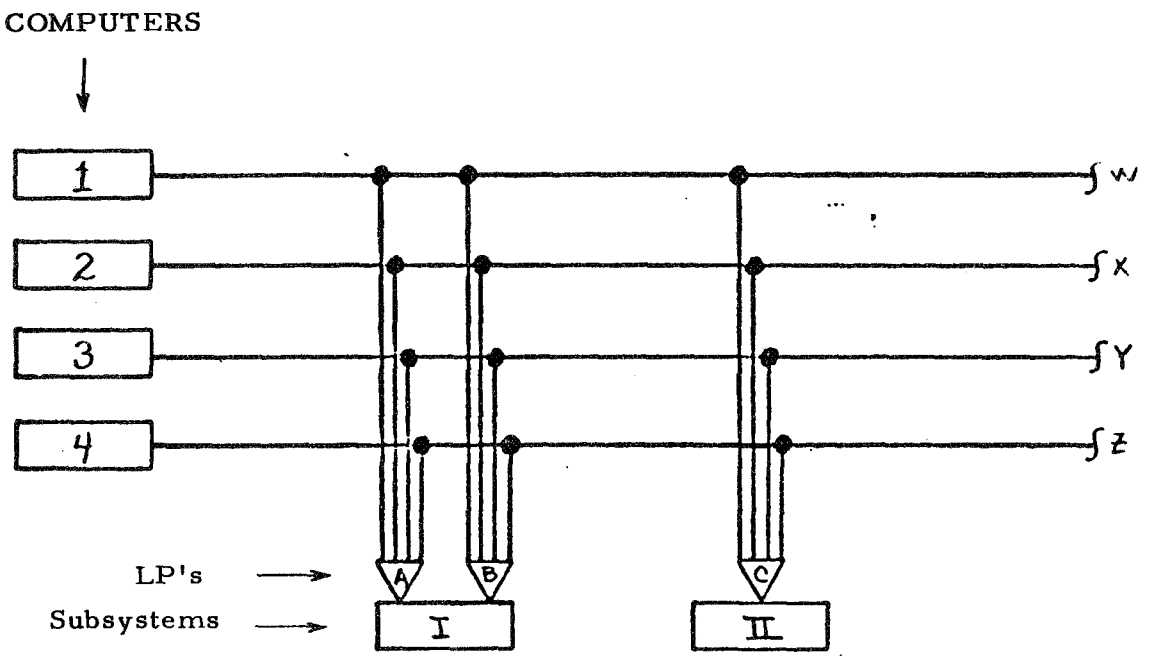


FIGURE 4-7. INPUTING VOTING EXAMPLE

4.3.2.3.2 External Failure - If the LP's in Subsystem I are being operated singly (para. 4.3.1.3) and a failure in LP-A unique to bus W occurs, then Computer 1 detects an error and is no longer capable of generating outputs for Subsystem II. The desired reconfiguration is to switch from LP-A to LP-B; however, switching requires the consent of a majority of computers. The only way that Computers 2, 3, and 4 can be informed of the problem is via Computer 1.

However, information from Computer 1 may be faulty due to a computer system failure, i.e., an internal fault may have caused Computer 1 to think that LP-A is failed. Hence, the other computers, seeing no error in LP-A themselves, disagree with #1 on the desirability of switching.

A possible solution is to design a more sophisticated reconfiguration algorithm. For example, if one computer votes to switch LP's, the other three could tentatively agree while remembering the fact that LP-A was apparently at least 3/4 operable. This would cause the desired switching to occur. Since all four computers would now presumably agree that the subsystem was operating, the failure could probably be attributed to LP-A. It is, of course, possible that the failure was sensitive to a particular mode, input data set, etc., and thus might disappear when LP is switched, but of course the reconfiguration algorithm could be designed to attempt to return to LP-A if subsequent failures in LP-B and/or Computer 1 are indicated.

If LP's A and B are being operated in parallel, a slightly different situation can occur. Again, assume that a failure occurs in LP-A which in unique to bus W. Now Computer 1 cannot select the correct data set (assuming comparison is the only available method), and therefore cannot generate outputs for Subsystem II. This time a clear ground rule violation is not apparent since one LP failure has occurred and Subsystem I in this configuration is only Fail-safe. Of course, input voting would have provided a distinct advantage in this situation since Computer 1 could continue to perform effectively even after the failure.

Once again, however, by complicating the reconfiguration technique the same benefit could be obtained, i.e., have Computer 1 try using each of its two data sets. When the output voter indicates agreement, the problem has been isolated and Computer 1 is still on the air.

4.3.2.3.3 Resolution Uncertainty - This case is only meaningful when the LP's are operated in parallel since it is assumed that, particularly with a synchronous bus system, the LP will be capable of insuring an identical data set on all four buses. Resolution uncertainty in data sets from parallel LP's must be resolved in each computer by identical algorithms (averaging, truncation, etc.). This would seem to present

4.3.2.3.3 (continued) - no particular problem until an error similar to the one previously described (para. 4.3.2.3.2) is postulated.

Now Computer 1 would derive a different value for the input and would cause an output voter discrepancy. Once again, this situation could apparently be resolved by increased software complexity. This time Computer 1 would have to tell the other computers that it has an input error from LP-A and request them to stop using data from LP-A so that all four computers could then use only LP-B and would once again agree.

It should be noted that techniques other than input voting are also available for isolation of bus and/or bus unique LP failures. One such technique would be transmission tests conducted by multiple computers, i.e., one computer requests, via a LP, that a test message be transmitted from another computer. Receipt of the correct reply validates both bus links, receipt of incorrect reply or failure to receive a reply indicate faults which can be potentially isolated by further tests conducted between other combinations of computers, busses, and LP's.

4.3.2.3.4 Summary - From the investigations thus far, it would appear that provision for input voting is not an absolute necessity, but its benefits in terms of simplification of fault isolation and reconfiguration procedures make it well worth considering. Further detailed design of these procedures will more clearly indicate the extent of the benefits to be derived.

4.3.2.4 LP Voting Mechanization - During the course of evaluating the various configurations, the actual mechanization of a voter at the LP gradually evolved. One of the first and most serious problems encountered was that of keeping the LP synchronized with the data. This is necessary to prevent the LP from voting on data from different computation cycles. For voting at the LP, it was assumed that the four IOPs are synchronized and consequently the busses are synchronized. This allows the LP to compare the data as it is received from the bus.

To meet the FOOS requirement with just four computers, an adaptive voter is necessary. But to account for reconfiguration the adaptive voter must be able to return to a previously failed computer/bus. It therefore seemed desirable that the voter be capable of being told which sets of data should be used for the vote, i.e., allow the computer system to direct the adaptation process.

Since the bus is susceptible to intermittent noise type failures, it was also desirable that the voter not adapt too soon. It is preferable to determine whether a failure is intermittent prior to initiating reconfiguration.

4.3.2.4 (continued) -

In order to provide maximum spare computing capability, triple redundant computations should be the maximum redundancy at any one time. The voter then is capable of looking at four sets of data but normally will only vote on at most, three sets at any given time.

The following mechanization attempts to account for the items mentioned above. The LP voter will initially be monitoring the four busses. During the computer program initialization the four computers will transmit to the LP's information about which busses should be used for voting. The LP will then "adapt" to those busses which the majority of the computers requested. The computer information used to direct the LP voter to a given set of busses will hereafter be referred to as the ballot.

The voter will continue to vote on three busses until a discrepancy is detected at which time the voter will direct the LP to use one of the two remaining valid sets of data and will report the discrepancy to all four computers. The computers will attempt to re-establish the "accused" computer. If the "accused" computer does not fail in the vote after being re-established, the failure will be assumed to have been intermittent. If the voter reports that it has failed after having been re-established, the backup computer will be brought on board and the ballot will be changed to adapt the voter to no longer vote on the failed computer/bus and also to ignore its ballot. After a second computer failure has caused the computers to reconfigure, the ballot could be changed to direct the voter back to a previously failed bus if it is again in use.

For R2 functions which are only double redundant, three busses could still be voted upon. In this case, the voter must be able to recognize and report absence of data as well as erroneous data. It would be left to the computers to determine if the absence was intentional or not. Also, for R2 functions, when a fault is detected, it cannot be isolated until the backup computer is brought on board. Then the three-bus voting will serve to identify the failed computer/bus so that it can be removed from the system.

The ballot should be transmitted with each block of data. The voter will not act upon any ballot that is not agreed upon by at least two computers. The voting results will be reported to all four computers, even though they may not be participating in the vote, so that each computer knows the status of the total computer system.

The capability of the adaptive voter to return to a previously failed computer/bus could be provided by a means other than the ballot method. The voter, whether implemented by hardware or software, could be mechanized to vote on three inputs and switch to a fourth whenever a failure is detected. In this manner the voter is not required to remember past failures, it always switches to whichever input is not being voted on at the time of detecting a failure. This technique will tolerate intermittent failures.

4.3.2.4 (continued) -

If the voter does not supply the results of the vote to the computer system, then all four computers are required to operate in a parallel, redundant manner. To allow some flexibility the voting results must be returned to the computer system. Suppose that critical functions are triple redundantly computed. When the voter detects a failure it notifies the RGC and switches its voting to include the spare computer. Since the spare computer is not performing the redundant computations, it will be voted out and the first failed computer will be switched into the vote again. If on this vote the first failed computer now passes, the computer system would assume that the failure was intermittent and would not reconfigure. If it fails the vote a second time, the computer system would assume a hard failure and would initiate configuring the spare computer to pick up the redundant computations for the next cycle. Interleaving double redundant with triple redundant computations would not be allowed with this technique because this could cause apparently simultaneous failures with only a single actual failure. Should this happen the voter would not be able to select the correct set of data.

4.3.2.5 Requirements for Reconfiguration - It is possible to operate any of the configurations in a manner which requires no reconfiguration within the computer system. This is accomplished by utilizing completely identical programs in all four computers which perform all critical functions (R1, R2, and R3) in parallel. Adaptive majority voting is performed on the four outputs at the LP/subsystem to allow FOOS operation. No feedback of voting status is required by the computers. Operating in this manner has the appeal of simplicity but has several major drawbacks:

1. Each computer must be sized (memory and speed) to be capable of performing the total operational task individually.
2. Restricted capability to perform non-critical functions since no "spare" computers are ever available.
3. Overall reliability penalty in terms of probability of success due to lack of flexibility in choice of reconfiguration paths.
4. Requirement that LP/subsystem account for resolution uncertainty in input data.
5. The LP's must be capable of adaptive majority voting.

The first level of reconfiguration capability which could be introduced into the computer system involves taking advantage of the varying function redundancy (error sensitivity) requirements. That is, R3 functions would be computed by only three of the four computers prior to a failure; R2 functions would be performed by only two computers at a time, and R1 functions would be split among the computers. This would allow reduction in the memory and speed requirements for each computer since R1 functions could be shared by two computers rather than being performed by both. It would further make available considerable computing capability which could be devoted to non-critical functions prior to the occurrence of failure.

4.3.2.5 (continued) -

In order to accomplish this mode of operation, two things must be available in the system.

1. The results of the voting process must be available to the computers.
2. The computers must be able to communicate with each other.

This level of reconfiguration will be performed on a function and/or a computer basis. All four computers will participate in the reconfiguration computations and the data that is transferred between computers will possibly include program modules as well as reconfiguration status, function initialization data, synchronization data, etc. The amount of data required to be transferred can potentially be quite large and hence the speed, efficiency, and convenience of the communication paths is significant. Procedures for reconfiguring each error sensitivity level of critical functions have been considered in some detail, but are only significant in this discussion in that they exhibit the extent of computer intercommunication required.

Another level of reconfiguration can be introduced if module-level switching (IOP/memory/CPU) under computer control is provided. The advantage of this is an increase in probability of success due to more flexibility in reconfiguration paths. The module-level switching would presumably be restricted to modules within a given compartment (two computers/compartment), however, the switching function must rely on a voting process in which all four computers participate. As in the previous discussion, the major significance of this level of reconfiguration is the necessity for a considerable amount of computer-to-computer communication; indeed, additional communication is required in this level since switching information and module level fault isolation data must be transferred. The additional fault isolation required to accomplish this type of reconfiguration does not seem germaine to the present problem (meeting FOOS) except for the implication of increased communication.

4.4 CANDIDATE CONFIGURATION EVALUATION

The following section presents the mechanization and evaluation of the nine configurations identified in Section 4.2.2.

4.4.1 Category 1

4.4.1.1 General Description - Each of the four data busses is dedicated to the IOP of one of the four computers. Communication between computers is only possible if multiple busses are connected to a given LP and the LP is capable of "relay" data transmission.

4.4.1.2 - Configuration 1A - In this configuration (Figure 4-8) only one bus is connected to any LP. A minimum of four LP's is required for each subsystem in order to meet the FOOS requirement which is in conflict with the intent of Ground rule #2.

4.4.1.3 - Configuration 1B - In this configuration all LP's are connected to all four busses (Figure 4-9). Adaptive majority voting is performed on computer outputs in each LP. No reconfiguration is required in the computer system to satisfy the FOOS requirement. If reconfiguration is employed to increase reliability and/or make available spare computational capability, each LP would report the results of its voting to the computers and the computers would use the LP's to provide computer-to-computer communication.

4.4.1.3.1 Input/Output Voting - Input voting is not possible in this configuration since data sets on a given bus are available to only one computer. Therefore, input errors will be detected as output discrepancies and further isolation will rely on such techniques as previously discussed (Para. 4.3.2.3). Output voting is performed in the LP's by a mechanism similar to the one described (Para. 4.3.2.4) if computer system reconfiguration is included. Simpler, less flexible voting mechanisms are of course possible if this level of reconfiguration is not included.

4.4.1.3.2 Reconfiguration - In order to allow any computer system reconfiguration, the results of the LP voting process must be transmitted to all computers and communication paths must be established between computers. The results of each LP vote could be readily transmitted in the "acknowledge" message issued in response to the computer's transmission request. Communication between computers can only occur with LP participation. Every LP would need the capability to relay data from one computer to another, which could be mechanized in the following manner:

For the case that one computer wants to transmit to another:

1. Computer 1 transmits to LP-A and specifies that the data is for Computer 2.
2. The LP would buffer the data and wait for a request from Computer 2.
3. Computer 2 would periodically address LP-A and request any data directed to it.
4. When addressed, LP-A would transmit the data to Computer 2.
5. The next time that Computer 1 addressed LP-A, the LP's acknowledge message would indicate completion of the relay.

Since the LP's buffering capability would presumably be limited, it would retain only one message at a time. If another computer requests data relay, the LP would indicate a "busy" status in the acknowledge message.

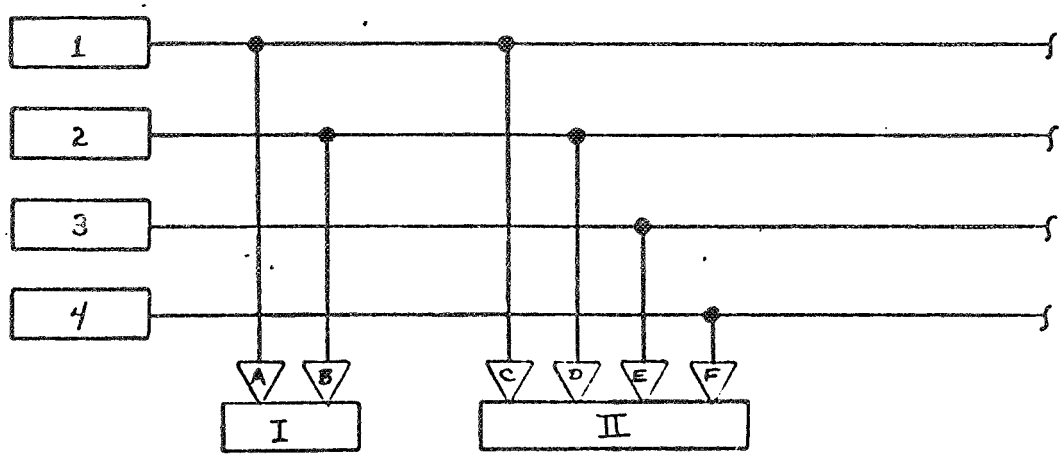


FIGURE 4-8. CONFIGURATION 1A

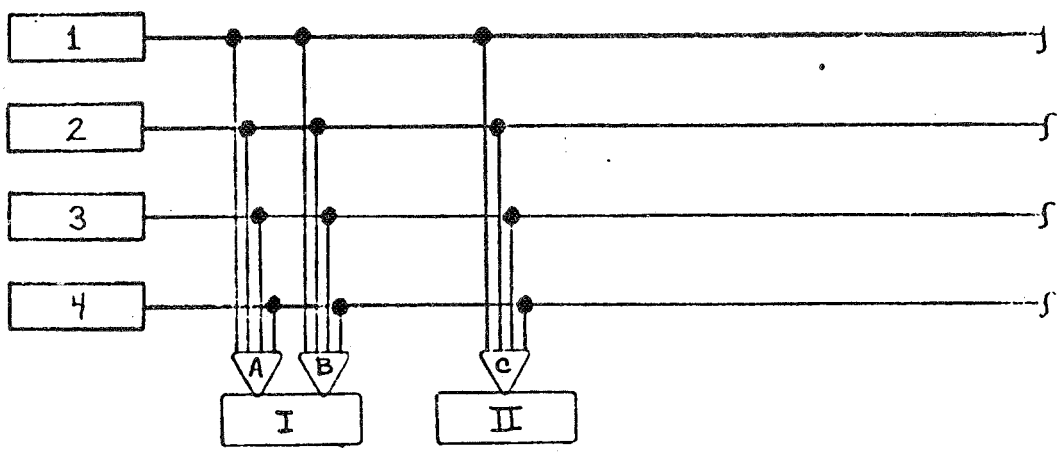


FIGURE 4-9. CONFIGURATION 1B

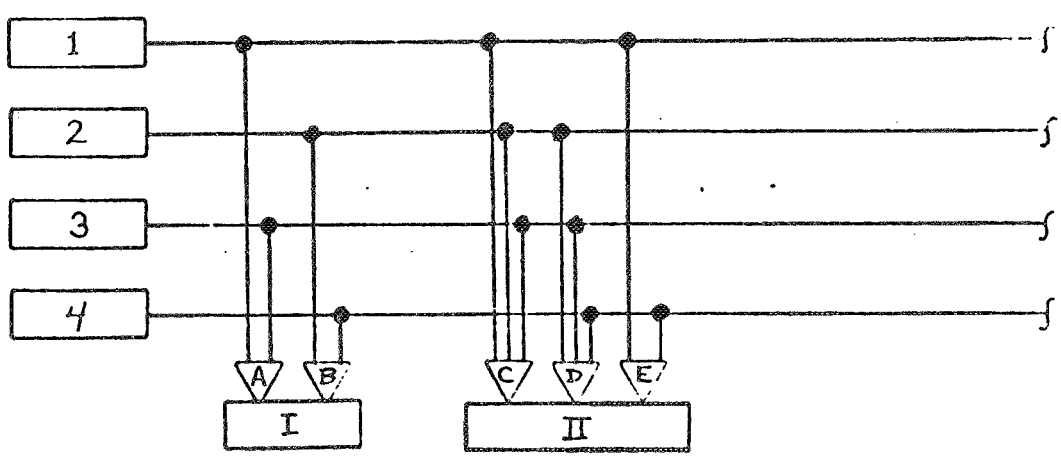


FIGURE 4-10. CONFIGURATION 1C

4.4.1.3.2 (continued) - A specific order could be established which assigned LP's as relays between particular computers such that failure of a given relay LP would automatically assign another one. However, computers would be required to send requests to multiple LP's on a regular basis in order to prevent undetected LP failures from breaking the relay link.

In the case that one computer wants to initiate a data relay from another, the relay system would function the same way except that the initiating computer would first relay a data request to the other computer.

4.4.1.3.3 Hardware/Software Mechanization Considerations - There seem to be no significant hardware considerations unique to this configuration. The bus/computer interface is the simplest of all the configurations and should therefore present the least problem from the standpoint of physical and electrical isolation of failures. However, a fair degree of intelligence is required of the LP in this configuration.

From a software standpoint, the computer intercommunication system is very cumbersome and inflexible, and makes high rate or high volume data transfer difficult.

4.4.1.4 Configuration 1C - This configuration differs from 1B only in that it allows connecting less than four busses to a given LP (Figure 4-10). There is little benefit to be derived from reducing bus/LP connections in this configuration since removing a bus from a LP also removes the capability of a computer. Therefore, in order to retain the FOOS capability the number of LP's in a given subsystem must be increased in direct proportion to the reduction in bus connections and any LP failure reduces the amount of computer system capability available to the remainder of the subsystem.

Operation of this configuration would not differ significantly from 1B, therefore no further discussion is presented.

4.4.2 Category 2

4.4.2.1 General Description - Each of the four computers can transmit on only one of the data busses, but each computer can receive data on all four busses. Communication between computers is therefore possible over the bus system (Figures 4-11 and 4-12).

4.4.2.2 Configuration 2A - This configuration allows only one bus connection for each LP. Since only one computer can transmit on a given bus, each subsystem must have at least four LP's in order to have a FOOS capability in the computer system. This is a rather undesirable restriction.

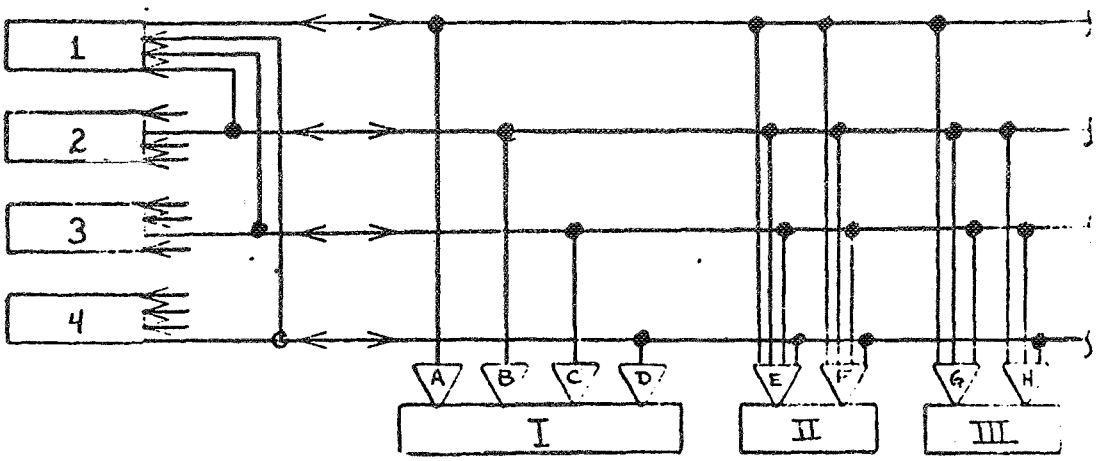


FIGURE 4-11. CONFIGURATIONS 2A, 2B, 2C

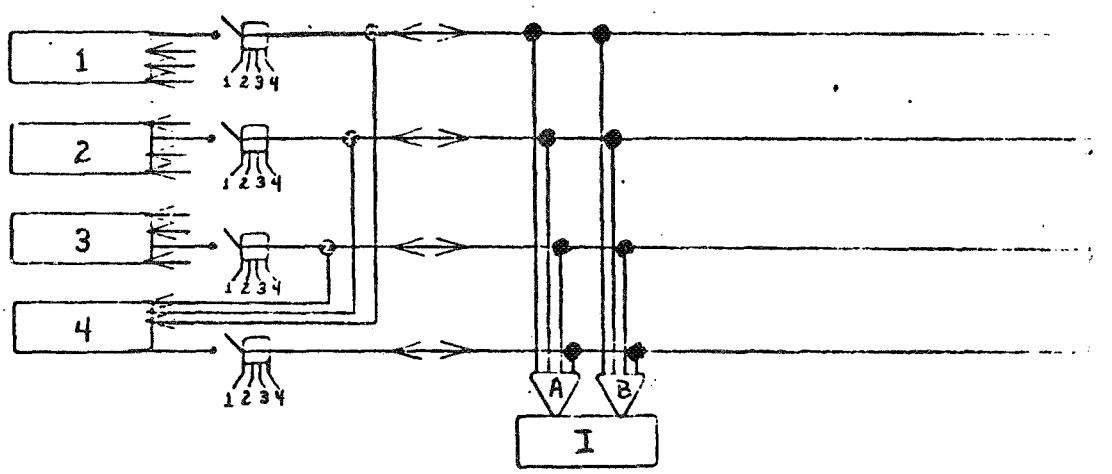


FIGURE 4-12. CONFIGURATION 2B WITH SWITCH

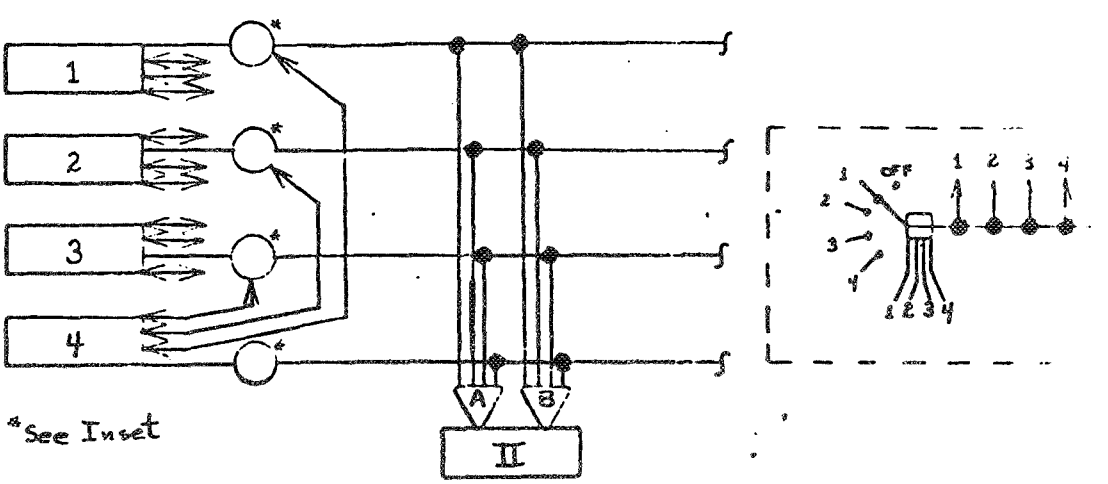


FIGURE 4-13. CATEGORY 3

4.4.2.2 (continued) -

Operation of this configuration does not differ significantly from configuration 2B which will be described in the following section.

4.4.2.3 Configuration 2B - Two alternate modes of operation were considered for this configuration:

Alternate 1 - Operation is identical to that described for configuration 1B except that input voting is performed and computers communicate directly over the bus in place of the LP relaying technique.

Alternate 2 - Each computer monitors the data being sent to the LP's from other computers. The data sets are then compared in each computer performing the associated function and the results of each computer's comparison is transmitted on its bus to the LP and/or used as a vote to control transmission on other busses.

4.4.2.3.1 Input/Output Voting - Input voting can be accomplished in this configuration since each computer can receive the data from all busses. The IOPs are assumed to be capable of selectively monitoring transmission from LP's by examining the address and control information in the data requests sent from computers to LPs. If the data set(s) being transmitted are being used by a given computer, its IOP will accumulate up to four copies of the data. All computers involved in the computations requiring the data will compare the redundant data and choose a single set. Each computer will then use that data set in its computations and will report the result of its comparison to the other three for use in the reconfiguration computations.

Output voting at the computer/bus interface (Para. 4.3.2.2.2) is performed in Alternate 2 of this configuration. An individual computer has the ability to monitor each of the other three busses, but for purposes of voting, it would only monitor that data which is redundant to its own calculations. Thus, if a particular computer is operating on R2 functions, it will only monitor one other bus for voting purposes. Since it has been assumed that the bus system is synchronized to within some tolerance time slot, the computer knows that it only needs to monitor the other bus(es) at the same time that its own data is being transmitted or received. If the redundant data does not appear on the other bus, the voter will fail that computer.

Once the vote has been taken, there are two methods of using the results. The first method is for the computer to transmit at the end of every data block, the result of the vote on that block. The LP would examine that vote and compare it with the votes received from the other busses and if a failure has occurred, switch to one of the busses which

4.4.2.3.1 (continued) - agreed with the majority vote. The LP is essentially voting on votes so that this process must also be adaptive, preferably by ballot (Para. 4.3.2.1) from the computer system. The second method of treating the vote could be used in conjunction with the above or autonomously. This method utilizes a switch on the bus (Figure 4-12). The majority vote is used in this case as a go/no-go indication to the switch which must be adaptive since it has four inputs. This provides a means of halting transmission of faulty data to a LP, which in turn recognizes the fault by noting the absence of data.

In the case of R2 data, a fault will be detected but only two voting results will be sent to the switch. In order to insure a majority-of-three decision by the switch, the backup computers for the R2 function will always send a "go" indication until one of two operational computers indicates a fault. When this happens, the backup computers will send a "no-go" indication taking both of the operational computers off their busses. One of the backup computers will now reconfigure to pick up this R2 function and when it is ready, it will notify the other computers which can now isolate the fault by comparison with the third set of data. Once isolated, the faulty unit will be taken off of the bus and the other two will provide the double redundancy required by R2 functions.

4.4.2.3.2 Reconfiguration - In Alternate 1, LP voting would be used for failure detection. The results of the voting process would be transmitted to all four computers over multiple busses. The reconfiguration process would then proceed utilizing the bus system for computer-to-computer communication without LP participation. Each computer would have a unique address analogous to a LP address, therefore, messages directed to a computer would not be monitored by LP's. Computer-to-computer messages would differ from computer-to-LP messages in that no acknowledge would be sent in response to a transmission request since the computer on the receiving end is incapable of transmitting on the same bus. Acknowledgement of data transfer would either be implicit in some other action or would be transmitted over the receiving computer's bus which would be monitored by the transmitting computer.

In Alternate 2, the results of the voting process in each computer will be exchanged with every other computer in order to compute the desired reconfiguration. Communication between computers would occur as described for Alternate 1.

Note that category 2 configurations allow a minor increase in reconfiguration flexibility over category 1. Since each computer can receive data on all busses, it is possible for a given computer to switch to another bus in the event of input-sensitive bus or LP failures which do not affect all busses.

4.4.2.3.3 Hardware/Software Mechanization Considerations - Two considerations seem significant from a hardware standpoint:

1. The implications of the additional bus-to-computer connections and communications.
2. The difficulty of implementing the bus switch pictured in Figure 4-12 to provide an adaptive voting function therein.

The only significant software consideration introduced by this candidate is the necessity for data comparison and voting in the computers. The effect of this load on the duty cycle requirements of the IOPs does not appear prohibitive.

4.4.2.4 Configuration 2C - The relationship of configuration 2C to 2B is analogous to the relationship of 1C to 1B as discussed previously (Para. 4.4.1.4).

4.4.3 Category 3

4.4.3.1 General Description - Each of the four computers can both transmit and receive on all four data busses (Figure 4-13). The proposed implementation of this category is common to configurations 3A, 3B and 3C, therefore, a single description will be presented. In this configuration, independent I/O channels from all four IOPs are routed to each of four switches associated with the four busses. The switch will connect at most one of the channels at a time to the bus allowing the computer associated with that channel to transmit and receive data over that bus. The switch position is determined by a majority vote on four inputs to the switch, one input from each of the computers. Further, each of the four computers can monitor data on all four busses independently of whether they are switched into any bus.

4.4.3.2 Input/Output Voting - Input voting is possible in this configuration and would be accomplished in the same manner as described previously.

In this category the computer/bus configuration at any given time is determined by the states of four different switches, one switch dedicated to each bus. The results of each computer's vote must be sent to these switches. In configuration 2B with switches, a go/no-go indication was all that was required by the switch. In this case, the results of the vote must define which computer is to transmit on a given bus. Again, since four inputs are required by the switch, the inputs must be generated in an adaptive manner under control of the computers. Note that the switches may have an "off" position for isolation purposes.

In category 3, faulty data will be transmitted, but a valid data indicator can be sent as the final word of every data block in the following manner. The redundant computers will be selected for transmitting data on the bus

4.4.3.2 (continued) - in a predetermined order. Assume this order to be (1) operating, (2) backup A, (3) backup B. At the end of the data set, the three computers will tell the switch to go to backup A. Backup A will then append the valid data indicator to the data set completing the data transmission. In the event faulty data is detected, backup A will append an abort indication and immediately begin transmitting the data a second time. If backup A fails in such a way that it sends an abort indication when indeed the data was valid, then the operating computer and backup B will detect this error and immediately vote to switch to backup B which will retransmit the valid data. In this manner the LP can be assured of receiving a valid set of data. The switch itself would not be capable of generating a valid data indicator even if it fails.

The voting process in category 3 will account for R3, R2 and R1 functions. Again, with R2 and R1 functions, if a failure occurs, the computer will automatically remove their data from the bus system until the fault is isolated after a reconfiguration has taken place.

Category 3 provides the potential for varying the number of redundant transmissions of redundantly computed information while still maintaining FOOS. If the bus system is outside the FOOS boundary, or if means other than voting can be used for bus fault detection, transmission of triple redundant computations may be limited to a single bus by relying on other detection methods (error coding) to detect transmission errors. This requires allotting a second time slot for every block of data to allow for transmission of data from a second computer in the event that a fault is detected in the first transmission. Since a valid data indicator is part of every data block, the LP will abort faulty data and prepare to accept the retransmitted data. Note that this does not imply that only a single bus will be in use at any given time. Several busses may be in use simultaneously, but no requirement to redundantly transmit any set of data is present. Use of several busses is desirable to reduce the density required for data on the bus.

4.4.3.3 Reconfiguration - In this configuration, the results of the voting process internal to the computer system will be the primary means of failure detection. This information together with the results of bus error detection reported by the LP's will as usual be distributed to all four computers for use in reconfiguration computations. Once again, the bus system is used for the required computer intercommunication without the aid of LP's. In this configuration, acknowledgement of the transmission request could be made over the same bus but might not be a desirable mechanization.

Note that another level of reconfiguration flexibility has been introduced over the category 2 configurations. Since the bus/IOP interface can be switched, a computer can switch to another bus in the event of a bus failure and all four busses can still be used after computer failures.

4.4.3.4 Hardware/Software Mechanization Considerations - The switching unit which controls the connection between IOP's and busses requires a detailed hardware mechanization consideration. The functional diagram (Figure 4-13) is, of course, simple-minded and actual implementation of this concept is treated in detail later.

From a software standpoint this is the most sophisticated configuration, but the software design difficulty relative to the other categories appears to be largely a matter of degree.

4.4.3.5 Alternate Implementations - There are many possible approaches to implementing the voting and switching requirements presented in this configuration. One which merits consideration is to implement the data voting and switching element totally in hardware. Assuming a synchronous bus system, this would appear feasible and could relieve the requirements on the IOP hardware and software.

4.4.4 Summary

The evaluation of the nine configurations has tended to uncover similarities in alternate approaches rather than differences. It appears reasonable to consider further only the two approaches corresponding to configurations 2B (Alternate 1) and 3C. Configuration 1B appears to be a workable approach, but reconfiguration demands a flexible, efficient computer-to-computer communication system. Configuration 2B (Alternate 2) does not seem reasonable since little benefit, either in terms of reduction in LP requirements or in reconfiguration flexibility is derived.

Of the two preferred approaches, configuration 3C offers the optential of a highly flexible system and is the recommended system concept. Table 4-1 is a summary matrix comparing configurations 1B, 2B and 3C.

TABLE 4-1 SUMMARY MATRIX

Category	Input Voting	Output Voting	Flexibility	Advantages	Disadvantages
Conf. 1B	No	LP	Poor	a)Simple in concept. b)Fewer problems w/ physical/electrical failure isolation	a)Inter-computer communications cumbersone.
Conf. 2B	Yes	a) LP b)Computer	Poor	Alternate 1: a) & b) same as above c)Inter-computer communications simpler.	Alternate 2: a)Output voting at computer does not relieve voting at LP
Conf. 3C	Yes	a) LP b)Computer	Good	a)Highly flexible. b)Output voting at computer reduces complexity of LP	a)Complex design req'd for switching elements. b)Potentially more complex software

4.5 DEFINITION OF CANDIDATE COMPUTERS

4.5.1 Introduction

The previous two sections presented the failure detection/reconfiguration concepts and several computer system organizational concepts to satisfy the FOOS requirements. As noted in the last section, system concepts 3C and 2B were selected as offering the greatest potential. The concept 3C required an adaptive voter switch at the interface to each of the four busses. In the progress of the study the alternate implementation referred to in Section 4.4.3.5 was given further consideration and selected as the preferred implementation of system concept 3C. This alternate mechanization places the voting on data function directly in the device that interfaces with the bus (an additional benefit is to vote before the data is actually placed on the bus, thereby inhibiting the transmission of erroneous data). This device will hereafter be referred to as the VCS (Voter-Comparator-Switch).

There are many alternate internal computer organizations that can be used to mechanize the four-level redundant computer system, the simplest being four conventional single computers. This section presents the internal organizations chosen as candidates. It should be noted that all of these candidates must meet the FOOS requirements. This requirement, as discussed in the prior two sections, is proposed to be met by adaptive voting means that do not rely on peculiarities of the internal computer organization (other than that four independent computing elements be provided). Four basic computer organizations will be discussed below. Each of the organizations have been considered for implementation by two technology approaches, one involves current low risk technology while the other uses higher risk future technology. Further, these eight possibilities are subject to the two system concepts, with and without the VCS device (this corresponds to configurations 3C and 2B in Section 4.4).

This yields a total of 16 candidates that provide a quantitative output to the evaluation task (Section 5.0). The description of and derivation of data on the candidate computers is presented below:

4.5.2 Candidate Organizations

Four computer organizations have been applied as candidates, they consist of two multicomputers and two multiprocessors. Since the internal organization is not the basis for meeting the FOOS requirement, a great deal of effort was not placed on deriving internal organizations; the organizations used are representative of many used in past studies (Ref. 4-3 through 4-9). As a result of the computer requirements analysis performed in Section 3.0 it was decided that the basic (non-redundant) storage capacity shall be 32K words (32 bits) and the speed capacity shall be 500,000 adds/second (operand from memory).

4.5.2.1 Non-Modular Multicomputer - A block diagram of this organization is shown in Figure 4-14. It consists of four single computers physically located in two compartments of the Space Station. Each computer contains a processor, memory and I/O section. The characteristics of the processor and I/O section will be defined later (Section 4.5.3). Each memory in this organization will consist of 32K words of 32 bits + parity. Each computer will be non-expandable in terms of memory and non-modular for redundancy purposes (spare P, M, or I/O modules cannot be provided within each computer).

This organization operates as a set of four single computers. It may be used as a single computer with redundancy or in a multicomputer mode where four computers are solving the computational load in a non-redundant manner. It is also envisioned that a mix of both redundant single computer and multicomputer operations may be performed by this system.

In addition to a serial digital bus to the LP's, each I/O processor contains a parallel (16 bits) digital interface for high speed data transfer to a mass memory or the information management system. This parallel interface is not included in the FOOS requirements.

Two system concepts (1 and 2) are shown in Figure 4-14. Concept 2 contains the VCS device referred to earlier and corresponds to configuration 3C selected in Section 4.4 above. Concept 1 simply excludes the VCS and would correspond to configuration 2B of Section 4.4. Since the VCS device is quite unique, it was subject to a detailed investigation. Therefore, further discussion of the computer system operation with this device is deferred to Section 4.5.5.

4.5.2.2 Modular Multicomputer - As shown in Figure 4-15, this organization is similar to the previous one in that it consists of four single computers. It differs in that each computer is modular within itself by means of an expandable common bus. The solid line modules in Figure 4-15 depict operational modules while the dashed line modules indicate redundant spare modules. The memory modules will be 16K words. Up to 64K words of operational memory modules (4) may be used on the common bus. Only one processor and I/O module may be operational on the common bus. The number of spare M, P and I/O modules that may be provided are defined later in the preliminary hardware mechanization considerations (Section 4.5). The two system concepts are also reflected on this organization in a similar manner as for the prior organization.

This organization may be operated in a combination of redundant single computer and multicomputer modes as in the previous organization. In addition, it may be initially configured with a variable number of memory modules in each computer. In fact, the number of memory modules turned on can be varied depending on the computational load.

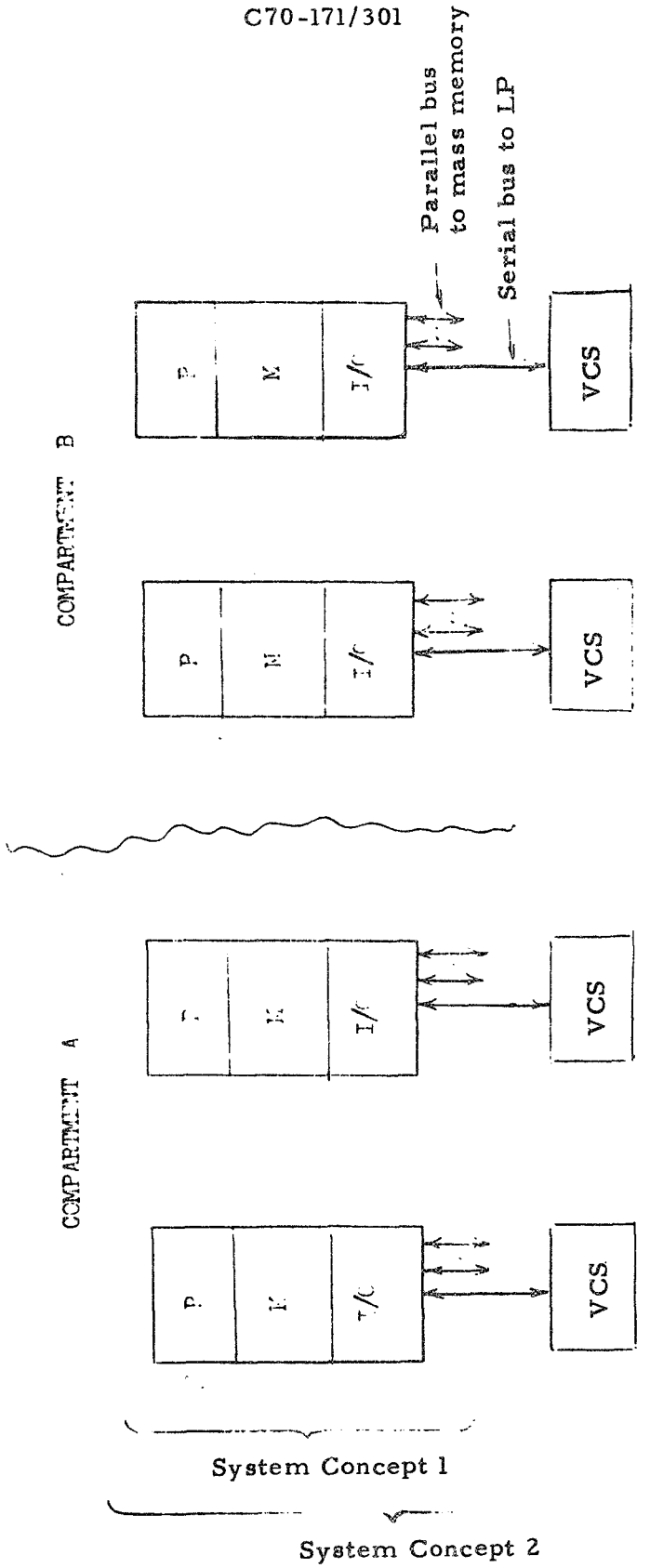


FIGURE 4-14. NON-MODULAR MULTICOMPUTER

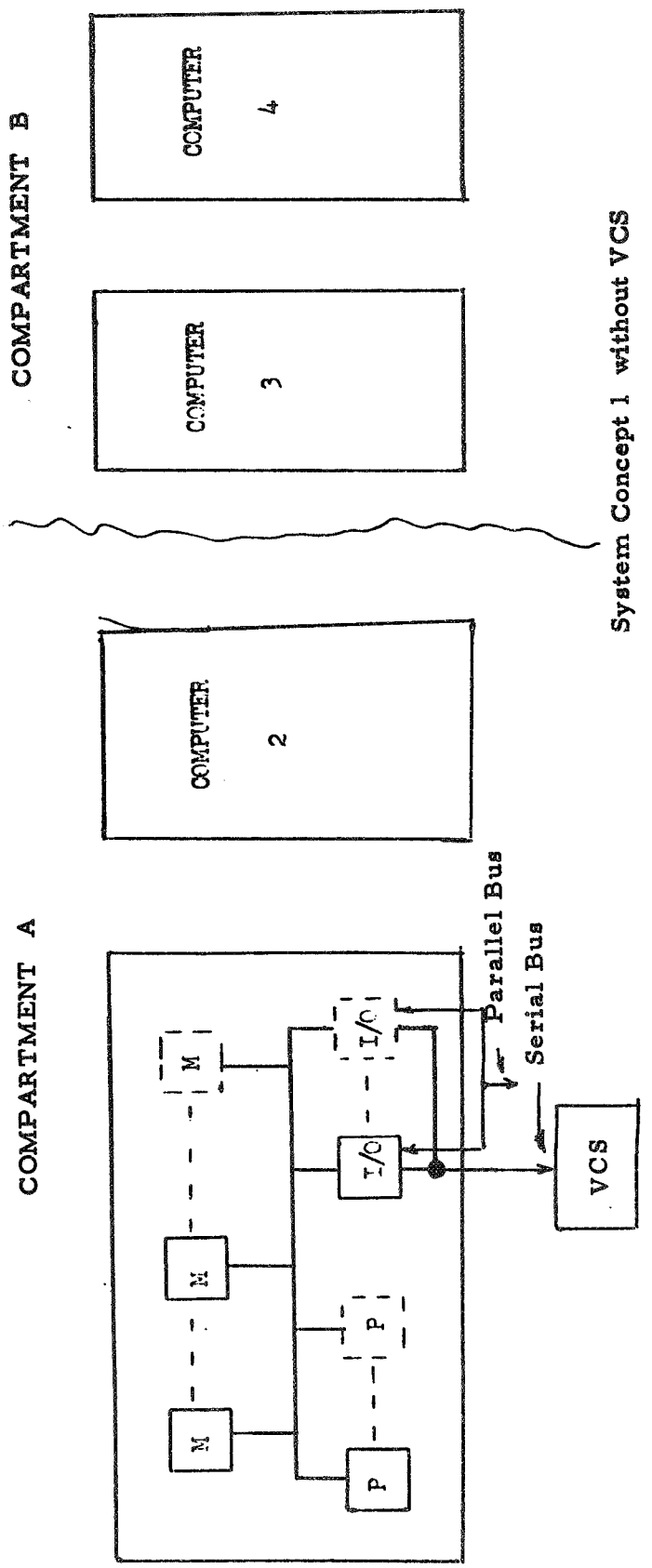


FIGURE 4-15. MODULAR MULTICOMPUTER ORGANIZATION

4.5.2.3 Non-Modular Multiprocessor - A non-modular multiprocessor organization is shown in Figure 4-16. Essentially, this organization consists of a pair of multiprocessors, one in each compartment. Each memory module has the capability of being accessed by two busses. A bus serves a processor and I/O unit.

The memory units shown in Figure 4-16 will contain 32K words of storage. The organization will not contain provisions for expandability or the addition of spare redundant modules. Several extra functions must be provided in this organization that are not required for the prior two organizations:

1. Each memory module must have two bus ports. The logic and hardware must be designed so that failures within the memory module cannot simultaneously render both busses useless. (See section 4.5.3 for some preliminary considerations). This is of extreme importance to preserve the FOOS capability.

2. Each memory module must have priority control logic to permit simultaneous operation of both busses.

3. A lockout function controlled by each bus is required. Two lines shall be included as part of each bus to control the lockout:

   00 No lockout - full multiprocessing

   01 No access by other bus - full lockout

   10 Set boundary registers to define scratchpad area for other bus (write area)

   11 Read only access by other bus

This organization may be operated as a pair of dual multiprocessors or in a variety of other modes. The provision of lockout functions in memory allow it to be set up to function as a multicomputer or redundant single computer as in the prior two organizations. This lockout provision allows the system to operate in a Fail Op-Fail Op-Fail Safe mode. (FOOS cannot be met when operating as a multiprocessor since independence failures could no longer be assumed.)

4.5.2.4 Modular Multiprocessor - As seen in Figure 4-17, this organization is similar to the prior multiprocessor. The difference is that a certain amount of modularity and expandability are provided. Each memory module will contain 16K words. The memory modules will be capable of operating with more than two busses to allow for processor expandability. For preliminary design purposes, provision for a total of three (3) busses should be provided. A total of 192K words of memory may be provided in each compartment. Spare redundant M, P, and I/O modules shall be capable of being provided in the system; the amount of this capability depends on initial preliminary hardware design considerations given in Section 4.5.7.

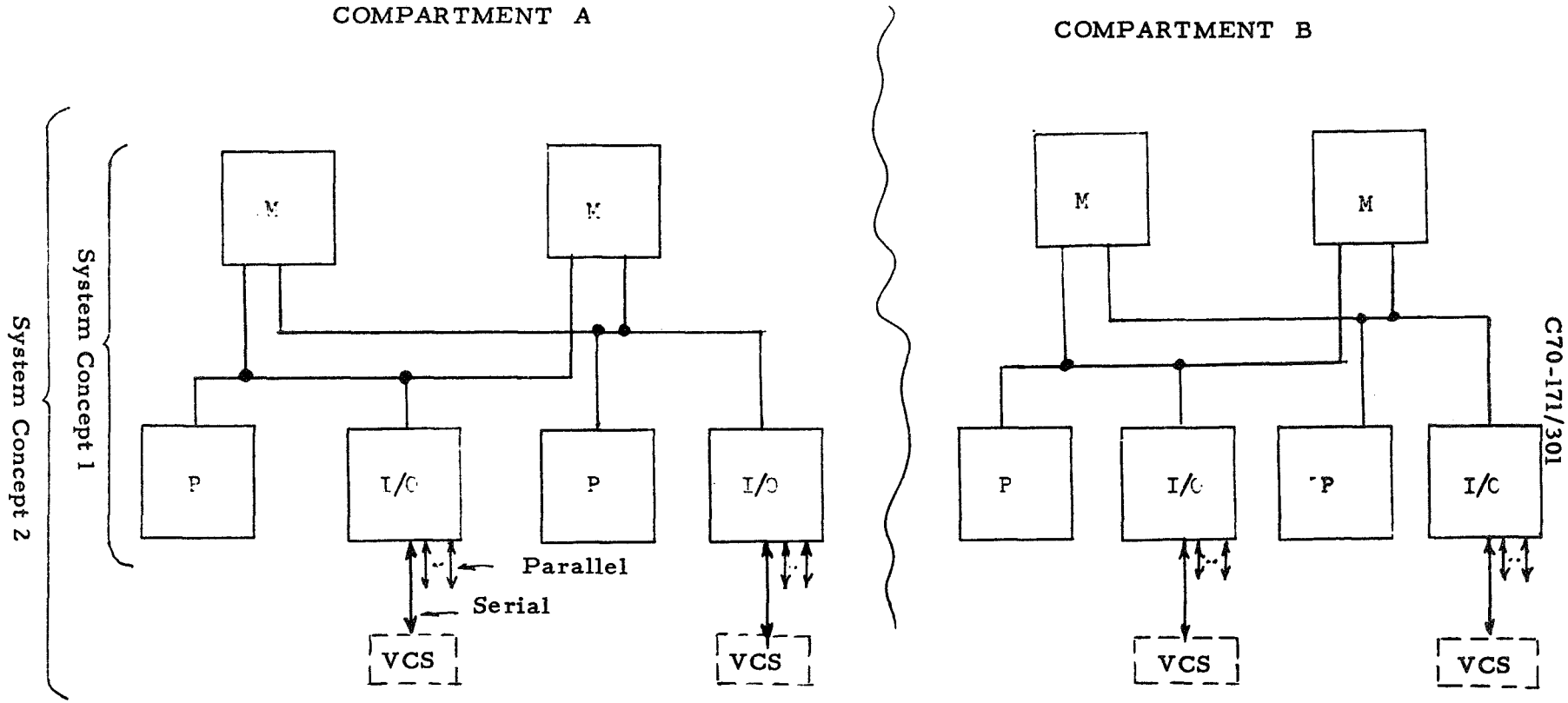


FIGURE 4-16. NON-MODULAR MULTIPROCESSOR ORGANIZATION

COMPARTMENT B
MULTIPROCESSOR 2
Same
as
Multiprocessor 1
COMPARTMENT A
Parallel
Serial
VCS
VCS
System Concept 1
System Concept 2

4.5.2.4 (continued) -

This organization will contain the same lockout hardware features in each memory module as described for the prior multiprocessor. It also may be operated in a variety of modes as previously discussed above.

4.5.3 Computer Architecture

The following ground rules apply to each organization. Differences in the M, P, and I/O units between each organization are due only to requirements unique to the particular organization. Once again, the details of the internal architecture are not the key to meeting the FOOS requirement, therefore, a representative architecture has been selected.

4.5.3.1 Processor -

- Arithmetic Word Length: 32 bits
- Data Options: 1. Fixed Point
  a) Half word
  b) Full word
  2. Floating Point
- Data Option Controlled by Mode Select
- Add Time (and operand fetch from memory)
  $\leq$2 μseconds
- Microprogram Control Unit
- Instruction Format: 16 bit and 32 bit
- Registers:
  32 bit
  - Accumulator
  - Lower Accumulator
  - Temporary Storage

  16 bit
  - 8 Base/Index
  - Program Counter
- Control Panel Interface
  - 4 Discretes
  - 4 Interrupts

4.5.3.2 Memory -

- Size specified in each organization
- Word Length: 32 bits + parity (1 parity bit for each 16 bits)
- Multiprocessor Interface: (as shown in Figure 4-18)

  The multibus interface is designed so that failures are independent among the interfaces.
- Multiprocessor lockout: (as specified in Section 4.5.2.3)

4.5.3.3 I/O -

- Independent Processor with limited I/O handling instruction repertoire
- Baseband time division multiplexed interface to bi-directional twisted pair bus (serial bus)
- 16 bit parallel high speed channel under external control

4.5.3.4 Memory Bus -

- Common bus shared by Processor and IOP with priority resolved by processor.

4.5.4 Computer Technology

Two approaches have been considered; the first is low risk state-of-the-art and the second higher risk currently developmental technology:

1. Logic: P channel 4 Ø MOS

   Memory: Plated Wire

   Conventional Packaging

2. Logic: P channel 4 Ø MOS

   Memory: MNOS and MOS Read Write

   Packaging: Beam leaded uncased devices on ceramic substrates with the substrates assembled into large packages (approximately 1-1/2" x 1-1/2") and these packages mounted on conventional boards.

These technologies were discussed in Section 2.0 and will not be discussed any further in this section.

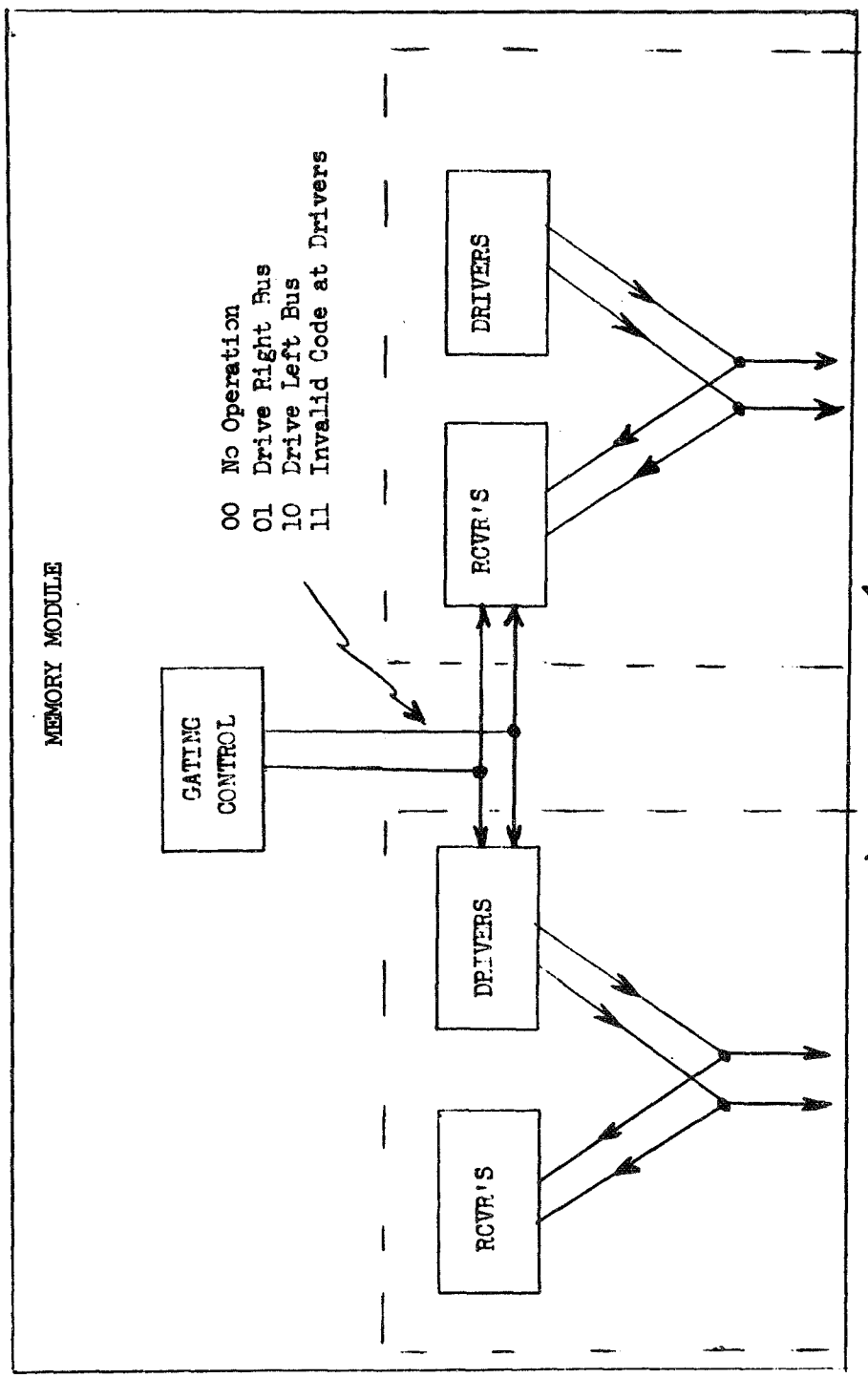


Figure 4-18. MEMORY MODULE INTERFACE FOR MULTIPROCESSOR

4.5.5 I/O Processor and VCS Mechanization

As discussed above system concept 3C uses a unique device entitled, the VCS (Voter-Comparator-Switch) that interfaces with the I/O Processors and the serial data busses to the LP's. This section will present a detailed investigation of the characteristics of the VCS device.

4.5.5.1 IOP-VCS Operation - This section describes the operation of the Input/Output processor and the voter-comparator-switch both within the computer and between the computers. Figure 4-19 shows the computer system with the interface between the computers and the I/O busses depicted. Four computers comprise the system, each is shown as being housed in one physical module. Each computer contains a connection to one of the four I/O busses in the system. In addition, each computer contains four other connections: three receive channels, one from each of the other computers, and one output channel to all the other computers; it is via these connections that the voter-comparator-switch (VCS) concept is mechanized.

Figure 4-20 shows the interconnection in greater detail where the VCS is depicted as a separate entity from Computer 1.

The architecture of the overall system has been designed so that the computer system may be operated in a wide variety of modes: four-way voting (all four computers doing the same job, with one or more of the VCS's voting on the information), three-way voting with the fourth computer dormant or doing a different job, two-way comparison with the other two computers also in comparison dormant or doing distinct jobs, and four non-redundant computers. The mode is under the control of the executive system which is distributed among all the computers. The executive is redundant, in a distributed sense, to satisfy the FOOS requirement. Actual control of the mode rests in the VCS device which essentially is the "front end" of each computer.

A detailed block diagram of the VCS - IOP section of a computer is shown in Figure 4-21. The VCS device outputs on one I/O bus; it has as inputs, the outputs of the four I/O processor sections of the four computers. As shown, these inputs to the VCS may be used by the voting, comparison or selection logic. The block diagram containing this logic is directed by a control unit as shown in Figure 4-21. This control unit is further directed by each of the four computers. The control unit is the heart of the VCS in that it must be adaptive to failures of the four computers. The control unit is designed to function under majority control of the computers.

The computers operate on their own independent clocks. While the clocks are nominally at the same frequency, there may be some drift between the computers. Hence, the computers cannot operate in bit sync.

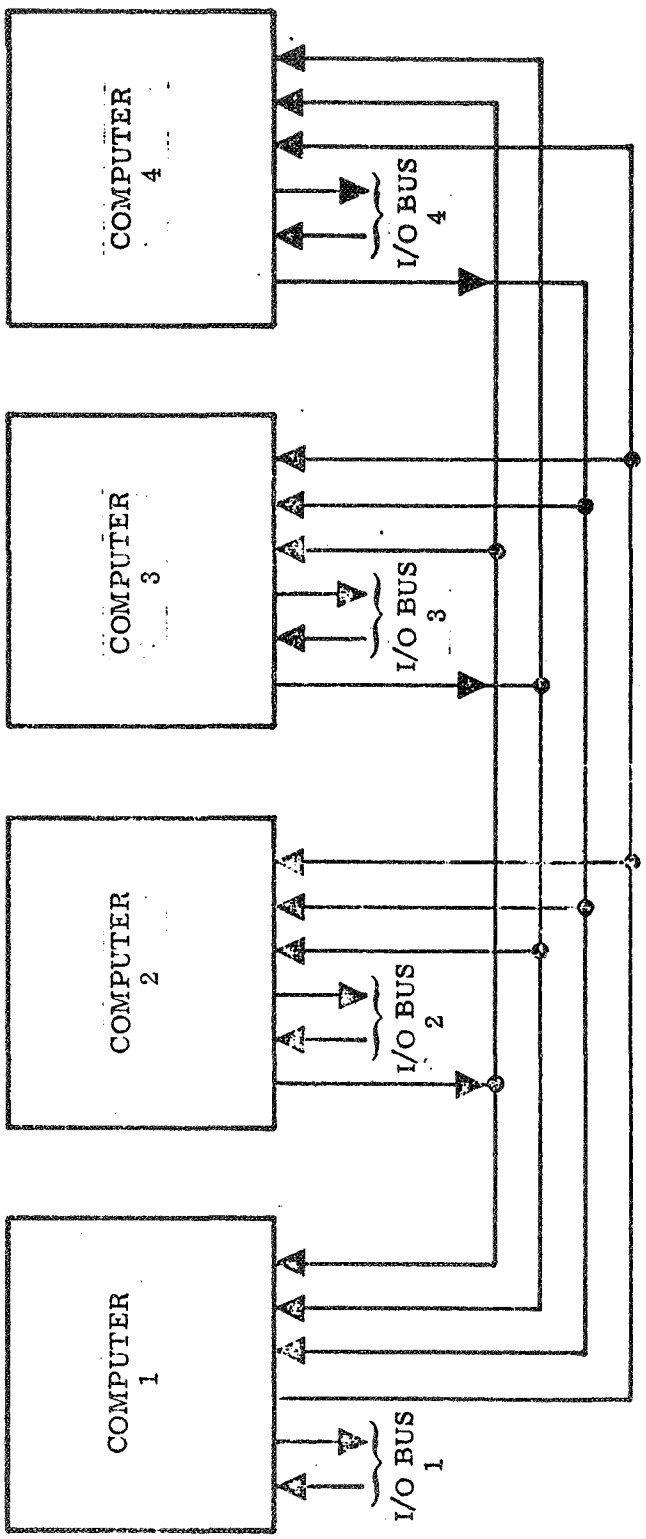


FIGURE 4-19. COMPUTER SYSTEM INTERCONNECTION DIAGRAM

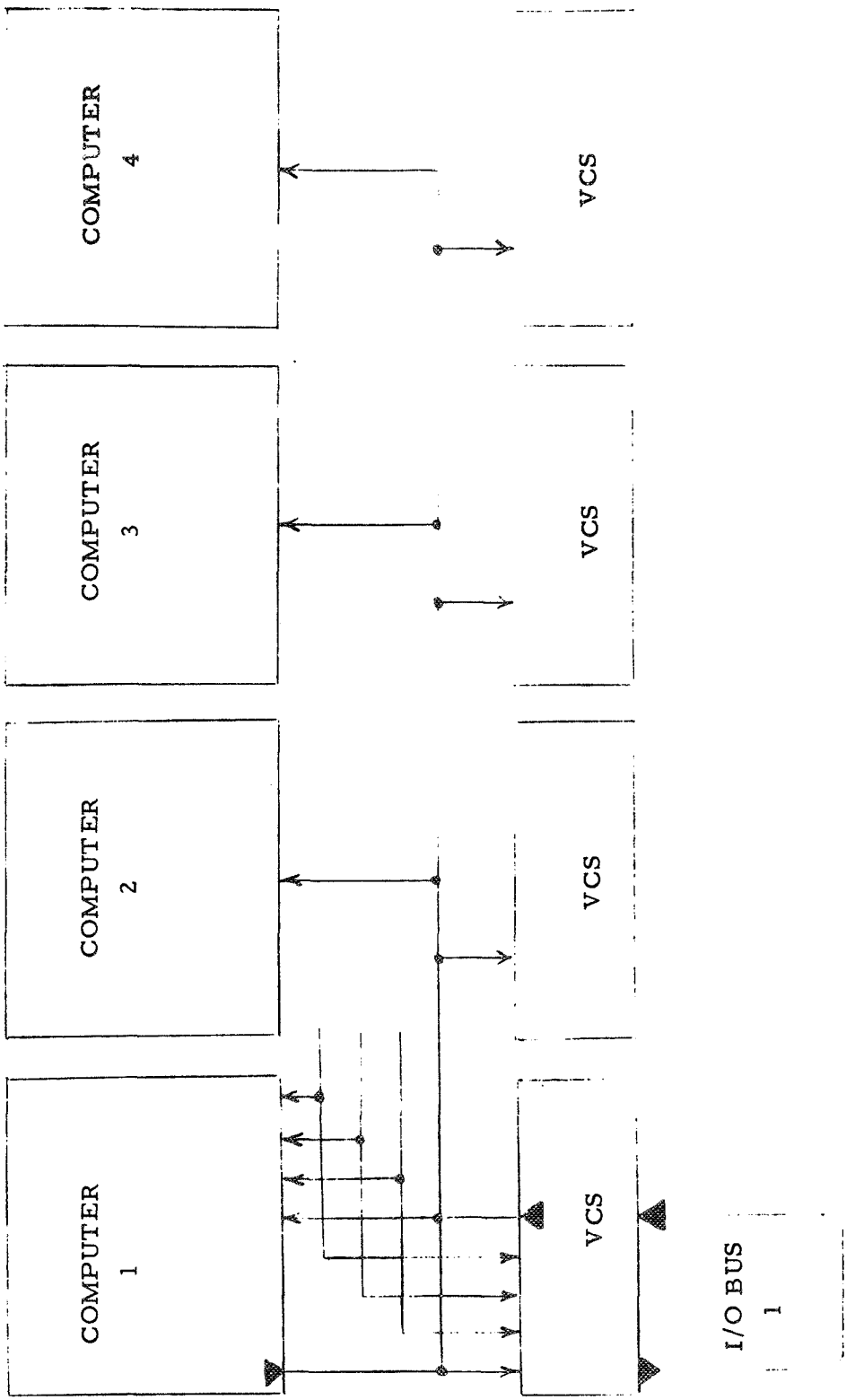


FIGURE 4-20. COMPUTER SYSTEM INTERCONNECTION MECHANIZATION

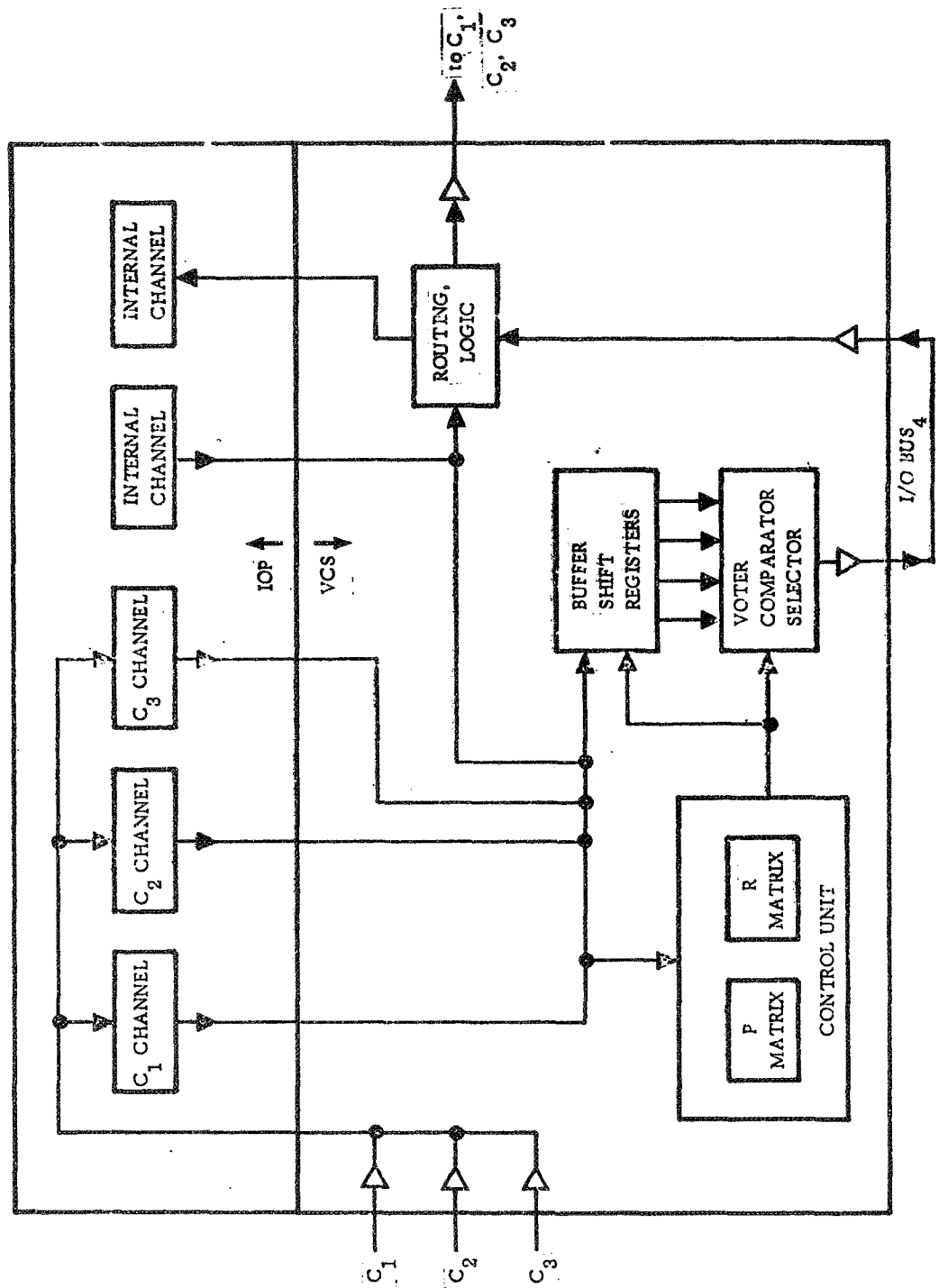


FIGURE 4-21. IOP - VCS MECHANIZATION

4.5.5.1 (continued) - (Sync of the clocks was considered, but rejected due to the difficulty of designing around the FOOS criteria). Relatively close sync of operation is required when operating in either the voting or comparison modes. Synchronization of the I/O processors (and the CPU) is accomplished by cross communication of the I/O processors. Each computer has a channel into the I/O processors of the other computers. When synchronization is about to take place, the computers to be synchronized transmit commands to each other via their respective channels. These commands are decoded and act as interrupts to the I/O processor. Receipt of the commands from the participating computers synchronizes the start of the I/O program. I/O processors will have the capability to mask the synchronizing interrupts thereby preventing failed computers from affecting other computers. The design of the VCS allows a ±1/2 word (16 bit) out of sync operation of the computers. This is accomplished by the use of shift registers in each channel from the other computers at the input to the VCS logic. This tolerance on synchronization allows for certain contingencies such as allowance for parity errors in memory operation in addition to the slight drift of the four clocks and similar occurrences which would be extremely difficult if not impossible to design around if some tolerance were not available.

The channel buffers in the I/O processor are serial in/out and parallel out registers that are monitored by decoding logic. This logic decodes commands and routes the commands and data to the appropriate destination. A computer may communicate to another computer directly via this channel. This buffer also sends commands destined for the VCS to the control unit of the VCS and data destined for the VCS is sent out to the shift registers. All commands and data will contain appropriate flag bits to be used in routing the transmissions.

Transmissions output by the VCS on the I/O bus are checked by monitoring the end of the bus. The decoding and routing logic associated with this channel sends the monitored information to the appropriate computers. Upon checking the transmission, the computers can send a validation to the LP's. In this manner errors on the bus may be detected and the LP's inhibited from using bad data.

The voting comparison and selector logic and the control unit of the VCS are described in detail in the next section. Basically, the control unit contains a "P" matrix and an "R" matrix. The "P" matrix is the permissible states matrix and the "R" matrix is the requested mode matrix. The "P" matrix basically is the set of failure indications of a computer on itself (each computer will contain hardware/software self test features capable of providing a self failure indication with a specific confidence, the self test/reconfiguration factors will be discussed separately in Section 4.5.6), and of a computers opinion of the other computers. (This is input to the control unit as a command with flag bits). The "P" matrix is driven by logic that operates under majority control. This majority control is adaptive to failures in the computer system.

4.5.5.1 (continued) -

The "R" matrix basically is the set of desired operational modes as input to the control unit from the computers (this is input as a command). The possible modes are a four way voter, three way voter, two way comparator, or a selector switch. The "R" matrix is connected to logic that is also adaptive to failures and operates by majority control.

The computers may change the mode of the system by sending appropriate commands to the VCS. Since majority control is required to do this, they must be synchronized and all be aware in their executive of the modes of the system that are to be used or they must communicate to each other via the IOP's of the desired modes of the system. In any case, the IOP's must all independently command the VCS of the desired mode of the system (the VCS adapts to failed computers by masking their commands).

The "P" and "R" matrix of the VCS may be monitored by any of the computers by commanding the VCS to send their contents to the computer. In this way a computer can check to see if the proper mode has been set up before outputting data.

It should be noted here that the initial considerations in design of the VCS considered using two buffer shift registers per computer input channel that provided inputs to the VCS logic. These dual shift registers allowed a maximum out of synchronism of 1 word in the outputs of the computers. Difficulty arises however if a failure occurs that results in a failed computer being up to 1 word out of synchronism (faster) then the non-failed computers. To cope with this situation, a third shift register was added to each channel. The maximum tolerance on synchronism remains 1 word. If a computer fails, such that it is less than 1 word out of synchronism with the non-failed computers then it will not disrupt operation of the VCS. If a computer is more than 1 word out of synchronism with the majority then it is automatically defined as failed.

4.5.5.2 VCS Mechanization - The control unit and voter-comparator-selector logic will be described below. A block diagram of this portion of the VCS is shown in Figure 4-22.

4.5.5.2.1 P Matrix - The function of the VCS control unit is to connect the VCS output unit appropriately to receive data from the four computers. Another function of the control unit is to determine which computers are good or bad. The control unit consists of two basic functional elements: The P matrix and the R matrix. The P matrix contains infornation on the good and bad state of the four computers while the R matrix contains the desired operating mode of the four computers (4V, 3V, 2CO selector).

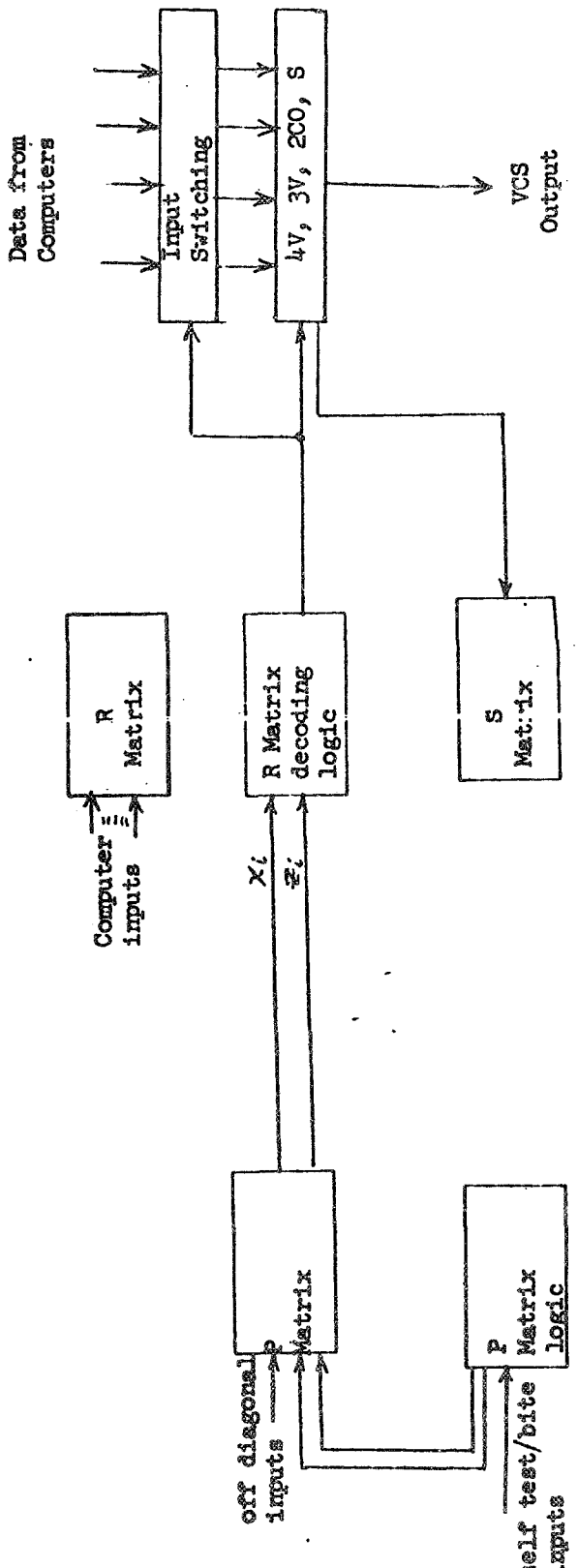


FIGURE 4-22. VCS IMPLEMENTATION

4.5.5.2.1 (continued) -

The P matrix is described below:

	A	B	C	D
A	AA	AB		
B				
C				
D				

It is seen to be a 4 X 4 matrix. The diagonal elements AA, BB, CC, DD are the prime information desired from the P matrix (this is what the R matrix uses). These elements define whether a computer is good or bad (if AA = 1, Computer A is good). The off diagonal elements AB, BA, etc., are one computer's opinion of another computer, i.e., AB is Computer A's opinion of Computer B. In general:

(i, j) = i's test of j
= 1 if i tests j to be good
= 0 if i tests j to be bad

The off diagonal elements (i, j) are directly input to the P matrix from the computers themselves while the diagonal elements are derived from logic associated with the P matrix.

The logic that derives the diagonal elements will be explained below. The basic criteria for declaring a computer good or bad is as follows: A computer is good until either it reports itself as bad (self test/bite signal from computer = 0) or a majority of the other good computers think it is bad. The equations that are used to derive the diagonal element will be given below. Those for computer "D" will be given; the equations for computers A, B and C follow directly.

The register that contains the "D" column of the P matrix, AD, BD, CD, DD, is as shown in Figure 4-23.

4.5.5.2.1 (continued)

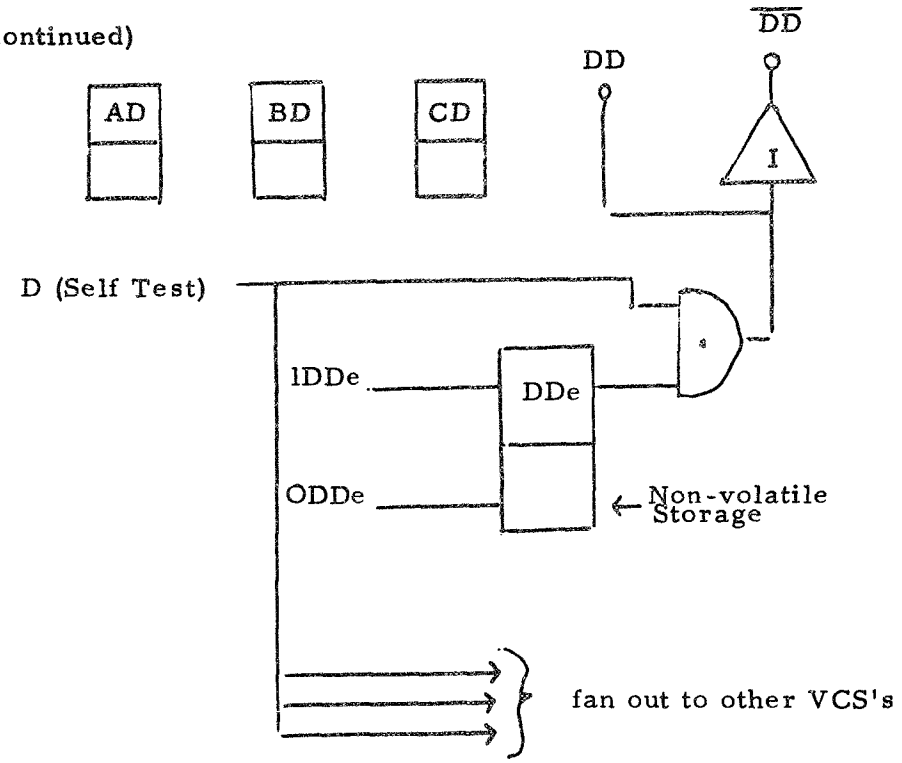


Figure 4-23. D Register Logic

The flip flops that contain the terms AD, BD, CD are set directly by each of the computers A, B and C respectively. The term DD is derived as shown in the diagram of Figure 4-23. The term "D" represents the self test/BITE signal received from computer D. This signal is "anded" with the contents of DDe, the enable DD flip flop. The one set and zero set (1DDe and ODDe) terms for DDe are given below:

$$\begin{aligned} ODDe = {} & (AA)\ (BB)\ (\overline{AD})\ (\overline{BD}) \\ & +(AA)\ (CC)\ (\overline{AD})\ (\overline{CD}) \\ & +(BB)\ (CC)\ (\overline{BD})\ (\overline{CD}) \end{aligned}$$

$$\begin{aligned} 1DDe = {} & (AD)(BD)(CD)(AA)(BB)(CC) \\ & +(BD)(CD)(\overline{AA})(BB)(CC) \\ & +(AD)(CD)(AA)(\overline{BB})(CC) \\ & +(AD)(BD)(AA)(BB)(\overline{CC}) \\ & +(AD)(AA)(\overline{BB})(\overline{CC}) \\ & +(BD)(\overline{AA})(BB)(\overline{CC}) \\ & +(CD)(\overline{AA})(\overline{BB})(CC) \\ & +(\overline{AA})(\overline{BB})(\overline{CC}) \end{aligned}$$

4.5.5.2.1 (continued) -

The ODDe term represents the adaptive logic of the P matrix. This allows the system to adapt or mask out failed computers from decision processes. Note that the DD flip flop must actually be non-volatile (it could be volatile if the data is also copied into a non-volatile storage, e.g., plated wire memory) storage since its condition cannot be lost due to any transients. This results since the adaptive logic can only handle single failures at a time and an attempt to re-establish the terms AA, BB, CC, DD after more than one failure can not be guaranteed.

The lDDe term shows the condition wherein the set of DDe is accomplished after repair of the system. Note that lDDe cannot be derived from the inverse of ODDe, an attempt to do so could cause the system to "blow up" after three failures. (After three failures the ODDe will go false and, if lDDe were derived from this, self test/BITE would have to be relied upon from all the failed computers) .

The output from the P matrix logic to the R matrix logic is the good/bad indication of the four computers. If we let

γ_i = good state of computer i

z_i = bad state of computer i

Then, for computer D

$$XD = DD$$
$$ZD = \overline{DD}$$

The situation when one computer fails, reports the other computers as bad, but fails to report itself as bad should be mentioned here. Suppose this has happened to computer A, the resultant configuration of the P matrix is:

	A	B	C	D
A	1	0	0	0
B	BA	1	1	1
C	CA	1	1	1
D	DA	1	1	1

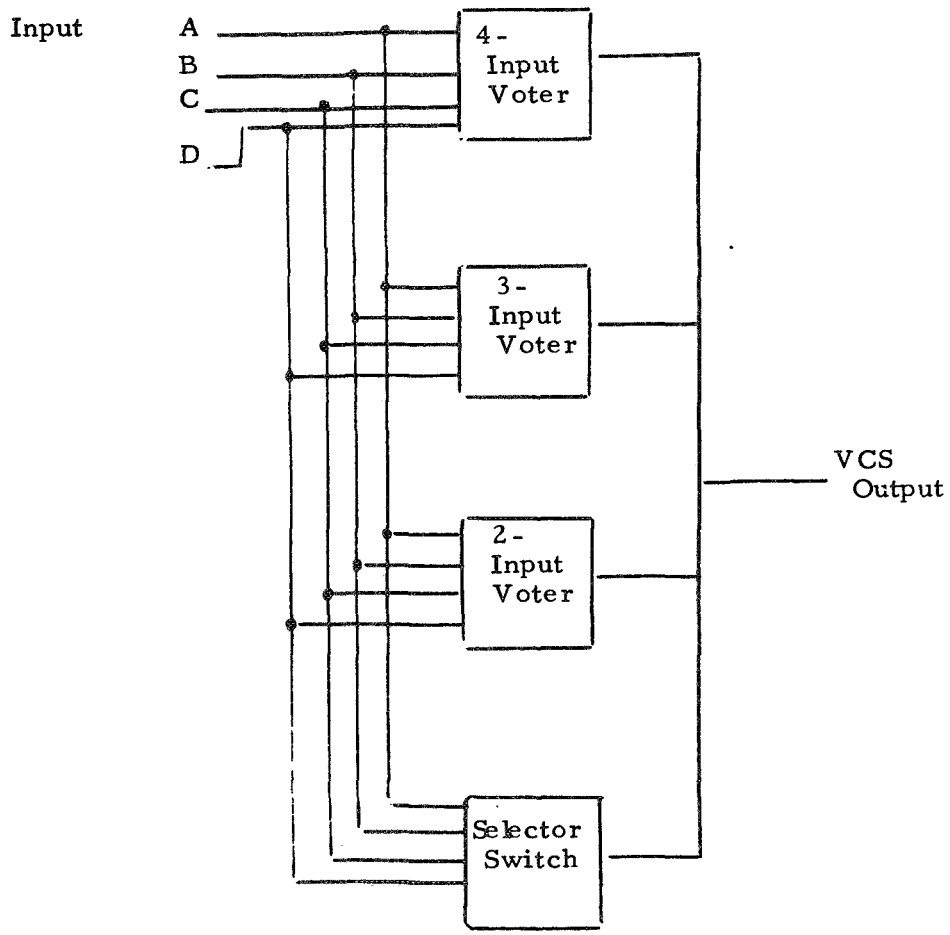
If this is the first failure in the system, then BA, CA and DA would be set to 1. The terms AB, AC, and AD being 0 would have no effect on the diagonal elements of the P matrix since the logic requires a majority opinion as explained above. Since a is the bad computer, it is up to computers B, C and D to insert 0's in BA, CA, and DA (only 2 out of the three are required); once this is accomplished, the term AA will be forced to a zero.

4.5.5.2.1 (continued) -

Note that after two failures, it may not be possible to reach a majority opinion in the logic associated with the P matrix. For the third failure, the primary term relied on is the self test/BITE indication.

4.5.5.2.2 R Matrix - The output unit of the VCS has the capability of acting as a 4 input voter, 3 input voter, 2 input comparator, or a selector switch on the outputs of the 4 sets of buffer triple shift registers as shown in Figure 4-24.

FIGURE 4-24 VOTER-COMPARATOR-SELECTOR



4.5.5.2.2 (continued)

The method of switching these inputs is by means of the R matrix. The R matrix switches the appropriate voter, comparator or selector to receive inputs from the selected computer(s). The R matrix coupled with the results of the P matrix (X_i, Z_i) then allows the VCS to function in a particular mode. For example, the R matrix would be set to:

	A	B	C	D
A	1	1	1	1
B	1	1	1	1
C	1	1	1	1
D	1	1	1	1

if the VCS is to work as a 4 input voter, as:

	A	B	C	D
A	1	1	1	
B	1	1	1	
C	1	1	1	
D				

if the VCS is to function as a 3 input voter between computers A, B, C.

No ambiguity should be presented by the R matrix, for example:

1	1	1	
1	1	1	
1	1	1	
			1

would represent a conflict to the particular VCS - that is, whether to operate as a 3 input voter on computers A, B, C or as a selector outputting computer D.

The R matrix decoding logic is designed such that the majority of the good computers (as defined by X_i, Z_i) must agree on a particular mode for that mode to be selected by the R matrix. A computer that will not be participating in that particular mode is required to insert all o's in its particular row. This essentially represents a don't care condition; i.e., it will go along with whatever mode the others want to operate in.

The R matrix is decoded as follows for a 4 input voter:

$$4V/ABCD = (rA_{15})(rB_{15})(rC_{15})\, X_A X_B X_C$$
$$+ (rA_{15})(rB_{15})(rD_{15})\, X_A X_B X_D$$
$$+ (rA_{15})(rC_{15})(rD_{15})\, X_A X_C X_D$$
$$+ (rB_{15})(rC_{15})(rD_{15})\, X_B X_C X_D$$

4.5.5.2.2 (continued) -

Where the nomenclature here is:

r_{ij}; i, j = A, B, C, D representing an element in the R matrix

r_{ik}; i = A, B, C, D k = numeric $0 \rightarrow 15$
representing decoded condition of ith row

A three input voter between computers A, B, C would be decoded as follows:

$$3V/ABC = (rA_{14})(rB_{14})(rC_{14})\ X_A X_B X_C + (rA_{14})(rB_{14})\ X_A X_B\ (rD_o + Z_D) + (rA_{14})(rC_{14})\ X_A X_C\ (rD_o + Z_D) + (rB_{14})(rC_{14})\ X_B X_C\ (rD_o + Z_D)$$

Similar conditions apply for 3V/ABD, 3V/ACD, 3V/BCD.

A two input comparator between A and B would be decoded as follows:

$$2CO/AB = (rA_{12})(rB_{12})\ X_A X_B\ (rC_o + Z_c) + (rA_{12})(rB_{12})\ X_A X_B\ (rD_o + Z_D)$$

Similar conditions apply to 2CO/AC, 2CO/AD, 2CO/BC, 2CO/BD, 2CO/CD.

A single input selector from A would be decoded as follows:

$$S(A) = (rA_8)\ X_A\ (rB_o + Z_B)(rC_o + Z_c)(rD_o + Z_D)$$

Similar conditions apply to S(B), S(C), S(D)

The additional proviso pertains to S(A) and that is X_A must be anded to the equation. The state of the R matrix must be decoded to obtain:

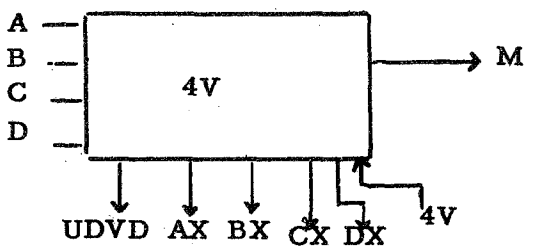
4V
3V (ABC)
3V (ABD)
3V (ACD)
3V (BCD)

2CO (AB)
2CO (AC)
2CO (AD)
2CO (BC)
2CO (BD)
2CO (CD)

4.5.5.2.2 (continued) -

S (A)
S (B)
S (C)
S (D)

These signals then enable the particular voter/comparator/switch and determine which lines (computers) are connected. The four input voter block diagram is:



Where

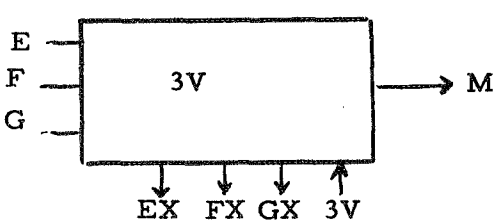
M is the voted output
AX, BX, CX, DX indicate a discrepancy in lines A, B, C, D respectively, UDVD indicates an undecidable voter discrepancy

$$M = ABC + ACD + ABD + BCD$$
$$AX = \bar{A}BCD + A\bar{B}\bar{C}\bar{D}$$
$$BX = A\bar{B}CD + \bar{A}B\bar{C}\bar{D}$$
$$CX = AB\bar{C}D + \bar{A}\bar{B}C\bar{D}$$
$$DX = ABC\bar{D} + \bar{A}\bar{B}\bar{C}D$$

$$UDVD = \bar{A}\bar{B}CD + \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + A\bar{B}\bar{C}D + A\bar{B}C\bar{D} + AB\bar{C}\bar{D}$$

where the terms A, B, C, D in the above equations are necessarily anded with 4V.

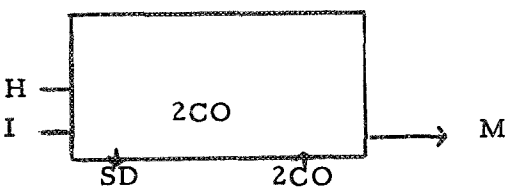
The three input voter becomes:



$$M = EF + EG + FG$$
$$EX = E\bar{F}\bar{G} + \bar{E}FG$$
$$FX = E\bar{F}G + \bar{E}F\bar{G}$$
$$GX = \bar{E}\bar{F}G + EF\bar{G}$$

4.5.5.2.2 (continued) -

And the two input comparator:



Where $M = H\bar{I}$
$SD = \bar{H}I + H\bar{I}$

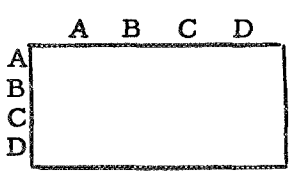
The switching of the input lines ABCD to the lines E, F, G, H, I is accomplished by means of the switching network:

$E = A.3V(ABC) + A.3V(ABD) + A.3V(ACD) + B.3V(BCD)$
$F = B.3V(ABC) + B.3V(ABD) + C.3V(ACD) + C.3V(BCD)$
$G = C.3V(ABC) + D.3V(ABD) + D.3V(ACD) + D.3V(BCD)$
$H = A.2CO(AB) + A.2CO(AC) + A.2CO(AD) + B.2CO(BC) + B.2CO(BD) + C.2CO(CD)$
$I = B.2CO(AB) + C.2CO(AC) + D.2CO(AD) + C.2CO(BC) + D.2CO(BD) + D.2CO(CD)$

and the selector is switched by

$M = A.S(A) + B\,S(B) + C\,S(C) + D\,S(D)$

4.5.5.2.3 S Matrix - The S matrix contains the error status of input data to the voting and comparison logic discussed above (4V, 3V, 2CO):



The S matrix is a 4 X 4 matrix, a 1 X 4 matrix is all that is required to indicate any errors in data from computers A, B, C or D. However, to allow the computers to reset the matrix to zero, the row (1 X 4) is repeated three times thereby resulting in a row for each computer. Therefore

$1, j = 2, j = 3, j = 4, j$
$1, j =$ any errors in data from computer j

For computer A, $j = 1$ and from the above discussion.

$$1, 1 = AX + UDVD + EX\,[3V(ABC) + 3V(ABD) + 3V(ACD)] + SD\,[2CO(AB) + 2CO(AC) + 2CO(AD)]$$

4.5.5.2.3 (continued) -

It can be seen that in a voting mode the logic will detect which computer disagreed with the majority, whereas in a comparison mode the logic will set two bits if a discrepancy exists.

4.5.5.2.4 VCS Communication - The VCS contains three matrices as described above. Each of these matrices may be sampled under command from the computers. All 16 bits of the matrix are sent to the computer if sampled. The computers can also set the matrices. However, the elements that they can set are limited.

P matrix: each computer can set the non diagonal elements in its row

R matrix: each computer can set its row

S matrix: each computer can reset its row

4.6 RECONFIGURATION ANALYSIS OF CANDIDATES

4.6.1 Introduction

Previous studies have resulted in two computer system configurations (Section 4.2) and for each configuration four computer internal organizations.

This section presents the evaluation of these eight configurations with respect to the general procedures used for fault detection and reconfiguration.

4.6.2 Computer System Level

4.6.2.1 General - At the computer system level there is little to be done in the area of fault detection and reconfiguration without discussing the computer internal organization. Consequently the discussion at this level will be brief. The two principal areas of interest are the use of voting on output data to detect failures and reconfiguration at the computer level to satisfy the FOOS criteria.

4.6.2.2 Output Data Voting - As indicated in previous sections, voting on independent, redundantly computed output data is the only method which will provide 100 percent confidence of detecting a failure. Self-test techniques do not provide 100 percent detection of failures. Given the two computer system configurations, the voting will be performed at the local processor in configuration 2B and at the computer in configuration 3C. The voter itself may be mechanized with either hardware or combined hardware/software techniques. In either case it is assumed that the voters are functionally the same.

When the voting is performed at the local processor (LP), as in configuration 2B, each LP will be voting and adapting independently of every other LP in the system. If the four computers are each executing

4.6.2.2 (continued) - identical programs in parallel, there is little difficulty posed by this situation. Each LP may switch to any one of the computers as required without affecting any other LP. If less than quadruple redundant computations are being performed then the computers must be notified of any failures detected by any one of the LPs. If all LPs report the same failure at the same time, a decision to use the fourth computer is obvious. If a failure is reported by less than all LPs, then several questions should be answered before reconfiguring the fourth computer. For instance, if only one LP voter reported a failure, this could imply that,

1. The voter failed.
2. There was an intermittent bus failure,
3. The computer failed such that it only affected the output data to that particular subsystem,
4. Or, if input voting is not performed, an LP/subsystem may have failed causing different input data to be transmitted to the different computers, thereby making the output data inconsistent.

To detect the voter failure, the computer would rely on the LP self-test, redundant LP feedbacks, or periodic tests by the computer, i.e., the computer would send data forcing all combinations of possible votes and examining the results from the LP. Situation 2, the intermittent bus failure, would be identified (if not by a bus self-test or error coding scheme) by requiring that a failure be detected at least twice sequentially before adapting the voter. In the third case where only data to a single LP was erroneous due to a computer failure, the computer self-test may detect this or a computer failure could be assumed once the bus and LP have been cleared of any fault. The fourth situation can be eliminated by input voting.

Only if the computer has failed would computer level reconfiguration be initiated. It is assumed that a "hard" bus failure would affect all LPs on that bus and would be indicated by all voters in the system receiving data during that program cycle. The reasons for investigating the nature of the failure before reconfiguring is to make maximum use of the fourth computer for non-critical computations before reconfiguration is required.

In configuration 3C voting would be at the computer interface, consequently, with proper isolation, bus and LP failures would not cause an erroneous vote on the output data. Whereas in configuration 2B the voter at the LP selects the correct set of data at the time of its receipt, in configuration 3C the voter selects the data prior to its transmission. When the computers have verified, via feedback, that the data was transmitted correctly they will validate the data for use by the LP. If a faulty transmission occurred, it could only have been the fault of the voter or bus. In this case the computers would select a second voter/bus and retransmit the data. The LPs will be aware of the situation by the absence of the valid data indication with the original set of data.

4.6.2.3 Computer Level Reconfiguration - The four computer system can be operated in two ways. One way is for all four computers to execute identical, complete programs in parallel. In this instance the voter determines which set(s) of data not to use. In configuration 2B the LPs will vote on and select the data sets. In configuration 3C the computers will vote on and select the data sets.

The second way is for three computers to be executing identical programs in parallel while reserving the fourth computer as backup. This allows the fourth computer to be used for non-critical data computations prior to the first failure. (Based upon sensitivity levels (Section 4.2) some functions could be computed in fewer than three computers. Reconfiguration of these functions is analogous to reconfiguring triple redundantly computed functions.) In this mode of operation the voter detects the failure, directs the use of one of the remaining two good sets of data, and begins reconfiguration procedures for bringing the spare fourth computer on board. Since the fourth computer is required to satisfy the FOOS criteria, the critical programs must be resident in its memory. Other sources for loading the program, such as mass memory or another computer's memory, will not be used to configure the fourth computer after the first failure. These other sources may be used in reconfiguration procedures following subsequent failures since the FOOS criteria will already have been satisfied.

Assuming that the critical programs are resident in the memory of the fourth computer, the modifiable parameters will be the only data required to complete configuration of this computer. A modifiable parameter is any word in memory that is not constant during the course of executing the program. This set of parameters includes all external input data plus internal flags, codes, modified instructions and intermediate computational data that is calculated and saved by the program for use in any subsequent computation cycle. Some parameters may be modified during the course of a single cycle but are either not required for subsequent cycles or are reinitialized to a fixed value prior to any subsequent use. These parameters are not included in the set of modifiable parameters required to configure a spare computer. Configuring a spare computer is discussed later in more detail (Para.4.6.3.2.1)

Once the fourth computer is configured to perform the redundant computations the voter will be directed to adapt to include the fourth computer and exclude the failed computer. This method assumes that the minimum time between failures is greater than the time required to configure the fourth computer.

4.6.3 Computer Internal Organization Level

4.6.3.1 General - Four computer internal organizations will be considered. Two organizations are of the multicomputer type, the other two of the multiprocessor type (Section 4.3.2). In each of the two types

4.6.3.1 (continued) - of organizations, one version is non-modular, i.e., no spare modules are available to the computer system, and the other version is modular. Since the non-modular organizations are special cases of the modular organizations, they will be treated secondarily.

4.6.3.2 Modular Multicomputer - The primary characteristic of the modular multicomputer organization is the availability of spare modules (memories, processors, and IOP's) associated with each individual computer (Figures 4-25 and 4-26). The communication path between two computers in a compartment is the same as between compartments, namely, the serial bus system or the high speed bus.

The following discussion and Paragraphs 4.6.3.2.1 and 4.6.3.2.2 assume that system configuration 2B is the one implemented unless otherwise stated. The distinguishing characteristics of configuration 3C will be discussed (Paragraph 4.6.3.2.3) after configuration 2B.

In configuration 2B each bus to the system is dedicated to a single computer IOP. Consequently if that bus or the IOP(s) associated with that computer have failed, the computer and its associated modules are lost to the system until repairs are made. As mentioned previously (Paragraph 4.6.2.2), the LP's vote on the validity of the data being output by the system and report back to the computers. When a bus fails the LP's will be unable to notify the concerned computer directly. The current definition of bus operation is that the end of a data transmission to an LP would cause the LP to automatically transmit status information. Hence bus failures would be detected indirectly by lack of this response. Also, by requiring the LP's to report to each computer the status of all computers, the bus failure can be made known to the concerned computer indirectly by its monitoring of the other buses.

The most obvious change required after a failure has occurred is to adapt the voter. Upon detecting the failure, the voter will automatically notify the LP to use one of the consistent sets of data. But before switching the vote to a new set of output data several other items must be taken into account. If the failure was due to an intermittent problem, such as noise on the bus, then the next subsequent votes on that data set should pass and adapting the voter is undesirable. It is assumed in this case that the computers are voting on the validity of input data. If the noise affected input data, the computer would recognize this and use the good data that was transmitted on one of the other busses. In this manner, the parallel computations of the three computers will always use the same input data even though one of the busses may have experienced a failure. If input voting is not used, then any erroneous data, even intermittent errors, used by a particular computer could cause that computer to have erroneous data from that point on unless the other computers can somehow correct the situation. The other items to consider before adapting the voter is the state of the spare computer/bus. Until the spare computer is configured to perform the redundant computation there is no requirement to adapt the voter.

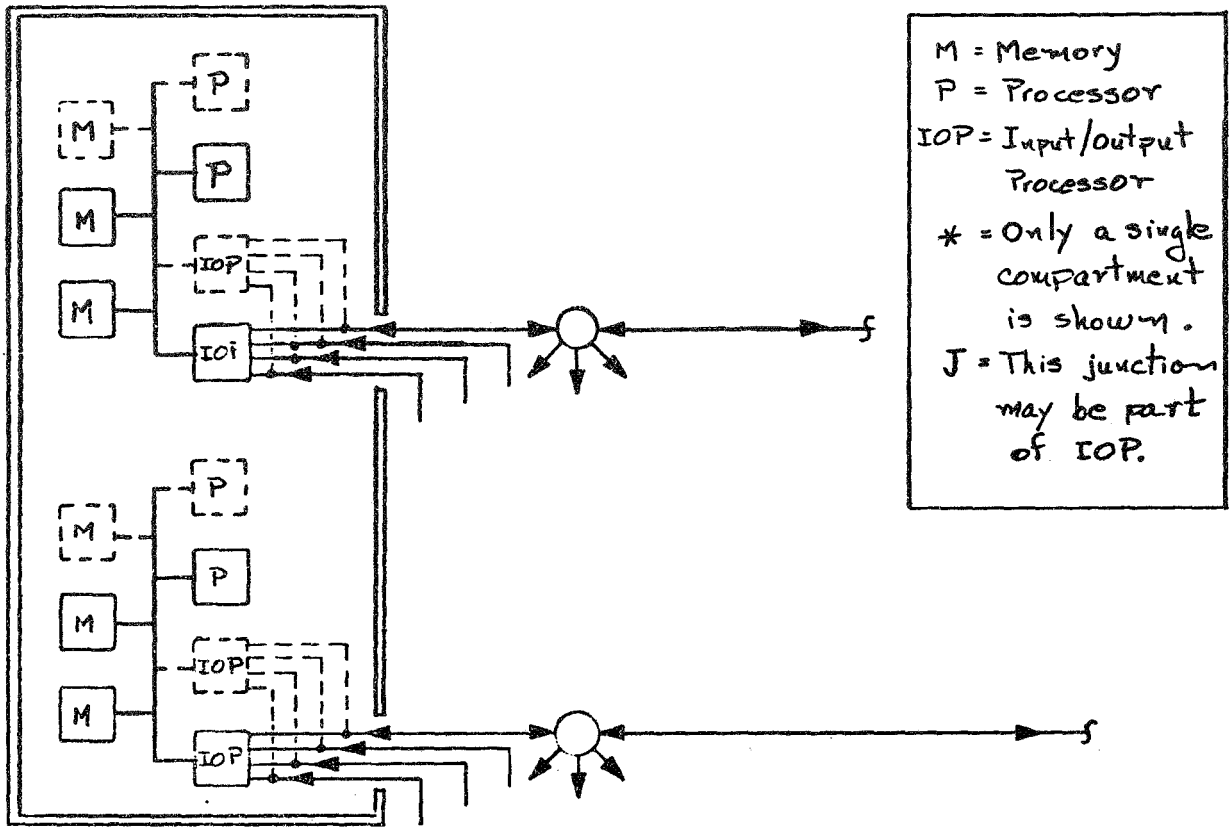


FIGURE 4-25. CONFIGURATION 2B* WITH MODULAR MULTICOMPUTER

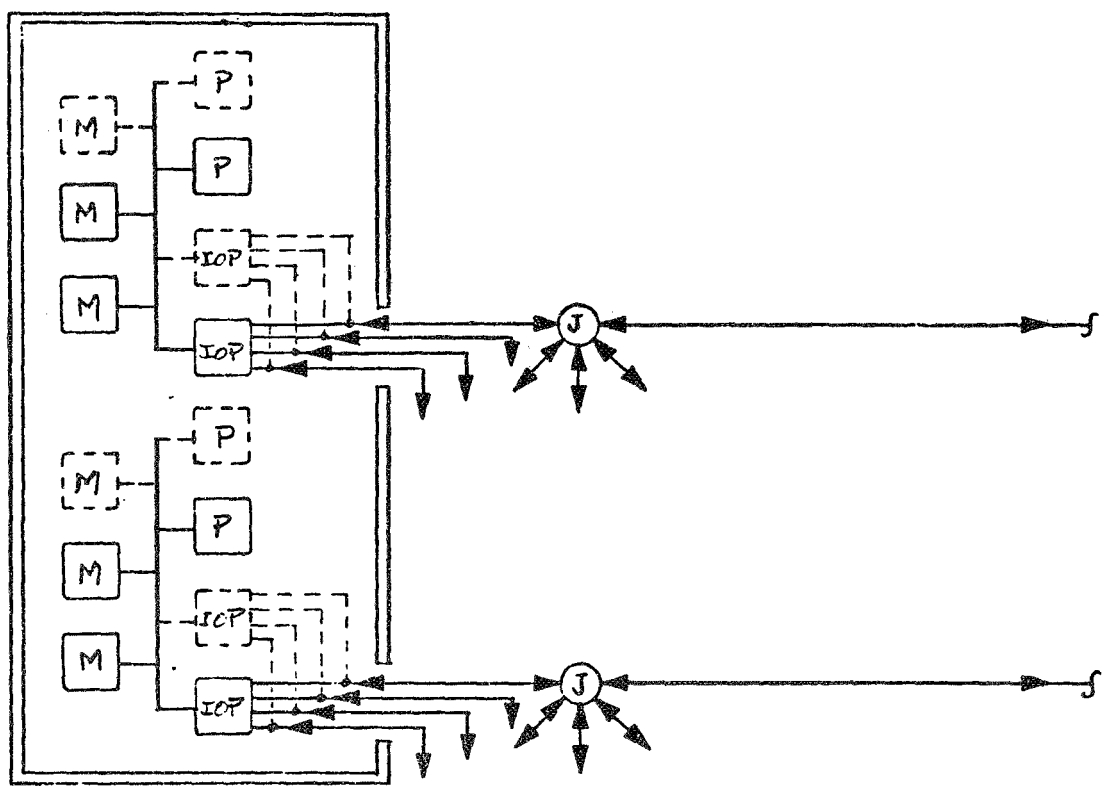


FIGURE 4-26. CONFIGURATION 3C* WITH MODULAR MULTICOMPUTER

4.6.3.2.1 Configuring a Spare Computer - Once the failure has been determined "hard", the next step to reconfiguring is to configure the spare computer, if there is one, to pick up the critical computations. It is assumed that the spare computer was idle or being used for non-critical computations at the time of the failure. The first task will be to load the spare computer's memory with the critical programs if they are not already resident in memory. Except when the computer is required to satisfy the FOOS ceiteria (Paragraph 4.6.2.3), the load source for the programs will be either the mass memory or the memory of one of the remaining good computers in the system. The former load method would be used when possible since the latter method would be more time consuming and would require participation by another computer.

As stated previously (Paragraph 4.6.2.3), the fourth computer is required to have all critical programs resident in its memory. As depicted (Figure 4-25) the modular multicomputer has two 16K memory modules for the critical programs and one 16K memory module for a spare. It is assumed that the spare 16K memory module would contain the non-critical programs and would be switched into the associated processor and IOP. The other two 16K memories would contain the critical programs and would be switched off or protected from any modification. The non-critical computations would include a routine to monitor the results of the voting being performed on the other three computers' outputs. When this routine determines that a failure has occurred it will switch in the memories containing the critical routines and will begin execution of a reconfiguration routine. This switching does not require the consent of the other computers since if it occurs inadvertantly or does not occur when required then that computer has failed and will be counted against the FOOS criteria. Also, the fact that it has or has not switched will not affect the other computers or LPs. If the computer being configured is not the fourth computer then it has already failed and probably has no spare memory available and must load the critical programs before proceeding.

Once the critical programs are in memory, the spare computer's reconfiguration routine will request transmission of all modifiable data (Paragraph 4.6.2.3) from the remaining good computer(s). This transfer of data is one of the more critical procedures required for reconfiguration. Two communication paths are available, the serial bus and the high speed bus. The high speed bus will normally be used for computer-to-computer communications.

If after the spare computer has been configured, the voter indicates that the spare is failing a second attempt to configure the spare will be attempted. However, this time the modifiable data will be transmitted via the serial bus network. (it is assumed that the high speed bus is not subject to the FOOS criteria and hence cannot be the sole means for configuring the fourth computer after the first failure. This switching to the serial bus, which is subject to FOOS criteria, will insure that satisfaction of FOOS is not jeopardized.)

4.6.3.2.1 (continued) -

Whichever bus network is used to communicate the data, since they are both subject to noise errors, it seems reasonable when configuring the spare computer, that the two remaining good computers transmit the required data in order that the spare computer could compare them for discrepancies. These would be resolved before reconfiguration could be completed. As previously pointed out (Paragraph 4.6.2.3), modifiable data which is not required as an input to any subsequent computations need not be transmitted from the other computers.

One technique frequently suggested for recovering after a failure is called "rollback". Rollback varies depending upon the system but in general it consists of retaining a known good set of data from one computation cycle so that if on the next cycle a failure should occur the programs, upon recovery will "rollback" to the good set of data to resume computations. This technique is not necessary in a system in which redundant computations are made. Given the ground rule that simultaneous failures will not occur, data from the current computation cycle are always available from one of the other computers. In addition, if one computer used rollback, all computers would have to rollback with it to maintain the consistent, redundant data required for voting.

If the amount of modifiable data required to "restart" a computation is so great that it cannot be transmitted during the period of the highest rate cycle, then the reconfiguration routine will decide which data for each rate will be transmitted in which cycle.

E.g., suppose that the routines calculate and/or sample data at three rates, 8, 4, and 2 times per second. Then, the calculations over a one second period for the data would occur as follows:

Cycle	Fraction of Second	8/sec(X)	4/sec(Y)	2/sec(Z)
1	1/8	X		Z
2	2/8	X	Y	
3	3/8	X		
4	4/8	X	Y	
5	5/8	X		Z
6	6/8	X	Y	
7	7/8	X		
8	8/8	X	Y	

Suppose that three cycles are required to send the total mount of modifiable data (X + Y + Z) to another computer. If the request for transmission occurs in cycle 2, then some or all of data set Z can be sent during that cycle. During cycle 3 the remainder, if any, of data set Z can be sent, but none of data sets X or Y can be sent since they

4.6.3.2.1 (continued) - will change in the next cycle which we have assumed is required to complete transmission. Consequently X and Y data sets must be sent during cycle 4. If the total of X and Y is too great for transmission during that one cycle then additional cycles will be required. In this example, if cycle 5 were required to complete transmission of X and Y we see that data set Z has changed again and consequently the first transmission is no longer useful. The reconfiguration routine will have to analyze the situation first and in this case would send all or part of data set Z in cycle 5, the remainder of Z and all or part of Y in cycle 6, and the remainder of Y and all of X in cycle 7. This sample also points out that sufficient spare transmission time for all of set X must be allowed in every 1/8 second cycle, for all of X and Y in every 1/4 second period, and for all of X, Y, and Z in every 1/2 second period.

Once the transmission of the set of modifiable data has been accomplished, the computer being configured will begin normal computations on the next cycle and the two remaining computers will now send a ballot to the voters to direct them to adapt to include the fourth computer's data and exclude the failed computer's data.

4.6.3.2.2 Reconfiguring a Failed Computer - After the spare computer has been configured the triple redundant computations are available to the system and further diagnosis, isolation, and reconfiguration of the previous failure can now proceed without interfering with the system operation. In most cases the failed computer can be performing self-test during the time that the fourth computer is being configured. In the modular multicomputer system, if one were to assume that a spare memory, processor, and I/O processor were available, then isolation could possibly be accomplished by substituting the spare modules, one-at-a-time, for the previously operating modules. This method has at least one shortcoming. Since the spare modules have not been involved in the triple redundant computations, it is possible that one or more of them have already failed, but have gone undetected, prior to the operational module failure. A self-test performed by software is effective but certain failures could render this means completely inoperative. Self-test of the failed computer by another computer is attractive but implies that the other computer can control the failed computer. If control by another is allowed it becomes possible for a failed computer to cause a failure in a good computer by erroneously exercising this control. This cannot be allowed. Majority voting by the remaining computers to control a given computer offers protection against this situation, but it introduces additional complexity which does not seem justified for self-test and diagnosis.

The self-test should be a combination of hardware and software techniques. Two principal hardware items are built-in-test (BIT) and a reconfiguration module. The BIT is normally a hardware check on the program execution based upon timing characteristics. BIT usually consists of a watchdog timer of a preset period (about 1 second) which operates

4.6.3.2.2 (continued) - independently of the computer. The program would be written such that it sends a specified code to the BIT at least once every period of the timer. If during any one timer period the code is not received, the BIT will indicate that an error has occurred.

The purpose of the reconfiguration module (RM) is to assure control of the reconfiguration process whenever the BIT indicates that a failure has caused the processor to lose control.

Upon system initialization the computer program will send information to the RM telling it which processor, memory, and IOP are available as spares. This information is essentially a pointer for the RM. The normal operation is then resumed by the active computer. If a failure now occurs which triggers the BIT error signal, indicating that the processor is no longer in control, the RM will automatically switch on the spare modules to which it was pointing and switch off all other modules. In addition it will indicate a starting location (probably by an interrupt signal) for the processor and memory just switched on. This starting location will be the beginning of a self-test and reconfiguration routine which must be resident in the spare memory. These routines would reset the BIT network and perform self-test of the newly configured computer. If this test passes then the modules operating at the time of the failure will be tested.

While testing the other modules the RM pointers will still indicate the original set of spare modules. This is to protect against the failed module bringing down the computer when it is switched on to be tested. If the failed module brings down the computer, the BIT timer will detect it and the RM will return to the original set of spares. By setting appropriate flags the self-test will know that the last module it switched on caused the failure which triggered the BIT. Self-test will be able to switch in memory modules one-at-a-time for testing but cannot switch on the processor since both processors cannot be on simultaneously. To test the processor the self-test will connect it with a good memory module not used by the self-test. It will then turn on the processor to be tested and turn its own processor off. (Note: Hardware is required to effect this simultaneous switching of processors.) The memory module being used for this test will have already been tested and loaded with a special routine by the spare processor. Again, if the BIT is triggered by this change, the RM will reconfigure the computer back to the original spare modules. If BIT is not triggered, then the special routine will test the processor further. If it is good then the spare memory (to which the RM is still pointing) will be connected. The program will return to the reconfiguration routine in the spare memory to test the IOP.

4.6.3.2.2 (continued) -

Once the failed module has been located, the remaining modules can be configured into a complete computer and the critical programs reloaded from either mass memory or another computer. The RM will also have its pointers directed by the program to any good modules which remain and which, if possible, are not used in the current configuration. The RM could also be directed to modules which are in use but this would only be done if no more spares are available. This reconfigured computer is now an operable spare and can be configured when needed in the manner described previously (Paragraph 4.6.3.2.1).

The presence of spare modules in a computer requires that a "resources table" be maintained. This table would list all modules available and as modules fail, the reconfiguration routine will remove them from the table so that subsequent reconfigurations will not use them. Since modules can be repaired or replaced by an operator, a means for him to update the resources table must be available.

If the failure was such that the computer is still grossly operable, such as an IOP failure which affects external inputs and outputs, the BIT will not detect the failure nor cause the RM to reconfigure. In this case the self-test portion of the normal program would be relied on to isolate the failure. The reconfiguration program would cause the failed module to be replaced by a spare, would update the available resources table, and report the new status to the other computers.

Reconfiguring a failed computer with a memory module that is not large enough for all critical programs is a possibility. This will not be considered at this time since it would be a much more difficult procedure and would only operate in a degraded mode.

In summary, the test diagnosis and reconfiguration internal to a given computer is autonomous and is a combination of hardware and software techniques using one-at-a-time replacement methods if necessary to determine the fault.

4.6.3.2.3 System Configuration 3C - As mentioned previously, the above discussions (Paragraphs 4.6.3.2.1 and 4.6.3.2.2) assumed that system configuration 2B was in use. If 3C (Figure 4-26) had been used there would be little difference. The major difference is that the voter is now at the computer instead of the LP and that a bus failure does not remove a computer from the system.

Again, input voting or a single source of input data is assumed to assure that discrepancies in output data are a computer fault. In configuration 2B every LP was a voter which posed a problem when all LPs did not agree on the vote. In configuration 3C this problem is reduced somewhat but not eliminated since voting is done on data sets and the failure may only involve a single data set rather than all data sets.

4.6.3.2.3 (continued) -

The primary advantage and reason for considering configuration 3C is the fact that it is the only one which can potentially reduce the number of busses required and consequently the voting complexity at the LP. It will be assumed for this discussion that the four computers are such that three are triple redundantly computing critical functions and are communicating with the system via a single bus. The fourth computer is used as a spare, is computing non-critical data, and is utilizing a different bus than the other three computers.

Initially the three comp uters are transmitting the redundant data to a voter which may be mechanized with hardware and/or software. The data is compared and a consistent data set is transmitted from the voter. Since the voter transmitter could fail it is necessary to verify the data on the bus. It is proposed that this be accomplished by making the bus loop, beginning and ending at the computer system, such that each of the three computers can examine the transmitted data and compare it to its own data. The voter will be contained in the computer, hence any failure of it constitutes a computer failure to be counted against the FOOS criteria.

Since a failed voter transmitter can send invalid data to the LPs a majority consent "valid data" indicator must also be sent. The LPs will not use the data until this indicator is received as the last word of the data block. If erroneous data was sent then the voter/transmitter or bus is at fault and the computers would switch to a second voter/transmitter and bus. The LPs will recognize the need to switch busses by the absence of the valid data indicator at the end of the data block. The computers will re-transmit the correct data on the second bus. Redundant data could be sent on a second bus in this configuration, but that defeats the main advantage of this configuration.

If a computer has failed then the voter will detect the discrepancy in the output data, switch to transmit one of the consistent sets of data, and report the failures back to the computer system.

When a computer failure is reported by the voter the reconfiguration of the spare computer takes place as previously described (Paragraph 4.6.3.2.1). When the spare computer has been configured, computer commands will cause the voter to adapt to switch out the failed computer (Paragraph 4.5.5) and begin voting on the spare computer's data.

If the failed computer happens to be the one performing the vote, then the computers would agree to switch to another computers's voter to leave the failed computer completely free to reconfigure internally.

4.6.3.3 Non-Modular Multicomputer - The non-modular multicomputer (Figure 4-27) consists of a processor, IOP, and a single memory module. If the four computers are used such that one is a spare prior to the first failure, then the operation is similar to that previously described (Paragraph 4.6.3.2.1). (Critical programs will be resident in the fourth (spare) computer, thus non-critical functions can only utilize the left-over memory.)

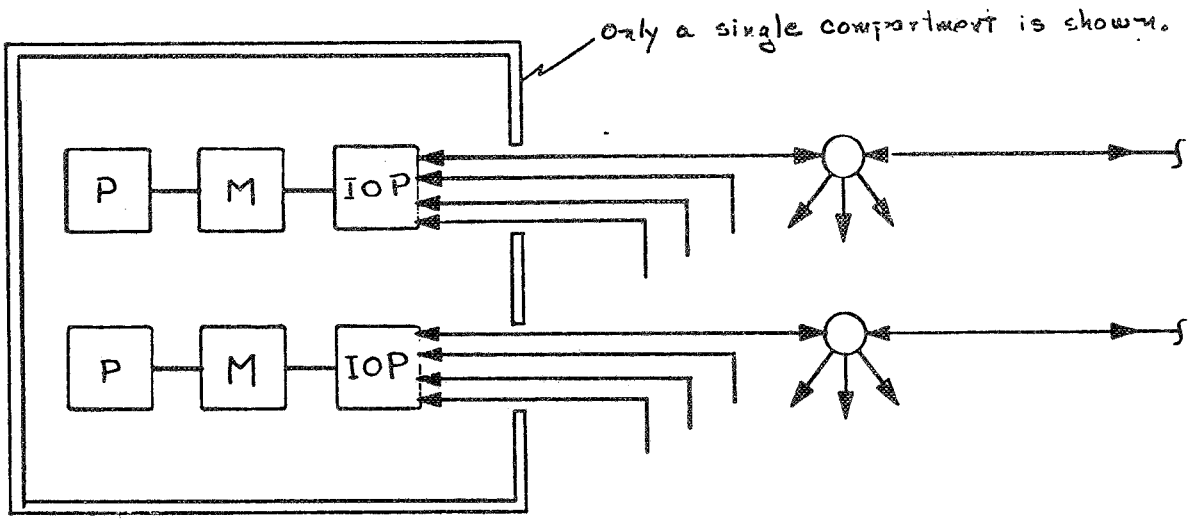


FIGURE 4-27. CONFIGURATION 2B WITH MULTICOMPUTER

The principal characteristic is that there is no computer internal reconfiguration possible. Once a computer has failed it is no longer available to the system, nor are any of its modules.

In the modular multicomputer the RM was necessary to switch spares into the system whenever the processor lost control and was unable to perform that function. Since the non-modular organization cannot have spares the RM is unnecessary. The BIT would still prove useful as an additional form of self-test. The BIT error signal could be used to turn power off to that computer or to inform the other computers or an operator of the failure. Self-test is required to isolate a fault when the system is in the safe condition. It will also furnish inportant diagnostic data for use when servicing the computer.

Since no reconfiguration is possible, the only routines required are those for configuring the fourth (spare) computer after the first failure.

4.6.3.4 Modular Multiprocessor - The modular multiprocessor organization combined with system configuration 3C is shown (Figure 4-28) with spare memories, a spare processor and a spare IOP. The spare modules represent the capability of the system to be expanded and are not necessarily suggested for the system. The following discussion assumes use of system configuration 2B except where noted and will assume spare modules to show the full capability of this system.

The multiprocessor differs from the multicomputer in that every memory module within the compartment is accessible to every processor and to every IOP in the compartment. Lockout protection must be provided to prevent one processor and/or IOP from interfering with the other processor and/or IOP operation and memory data. Since the FOOS criteria requires triple redundant computation, the operational use of the multiprocessor would be nearly identical to that of the multicomputer, and in fact does not fall under the normal definition of "multiprocessing."

In system configuration 2B, the voting will still be performed by the local processors. When a failure is detected, the fourth computer, again assuming that it was serving as a spare, will be configured in the same manner as described for the multicomputer organization (Paragraph 4.6.3.2.1). Even though one computer can access data directly from the adjacent computer's memory there is no advantage, since for reconfiguration, as was previously stated (Paragraph 4.6.3.2.1), two redundant sets of modifiable data will be requested and compared to detect transmission errors. This means that communications with the other compartment are required and are the same as for the multicomputer organization.

The computer internal fault diagnosis and isolation is also similar to the multicomputer operation (Paragraph 4.6.3.2.2). It is more complex because of the many different communication paths to be checked and because of the additional modules to be tested and their history recorded. If the normal self-test does not detect and/or isolate the fault, then substitution of spare modules may be attempted to determine the fault.

Whereas the modular multicomputer required a reconfiguration module, the second operating processor in the compartment will serve this function for the multiprocessor. Loss of control by a processor will still be indicated by a BIT network.

This control by one processor over the other introduces perhaps the most critical problem associated with the multiprocessor. The modules comprising two computers are not electrically isolated from each other as they are in the case of the multicomputer. Consequently, devising a means of protecting against a failed computer (processor, memory, IOP) from "failing" its adjacent computer is necessary as was indicated by the preliminary design considerations of Paragraph 4.5.2.3. For reconfiguration it is very desirable that software be able to control the "lockout."

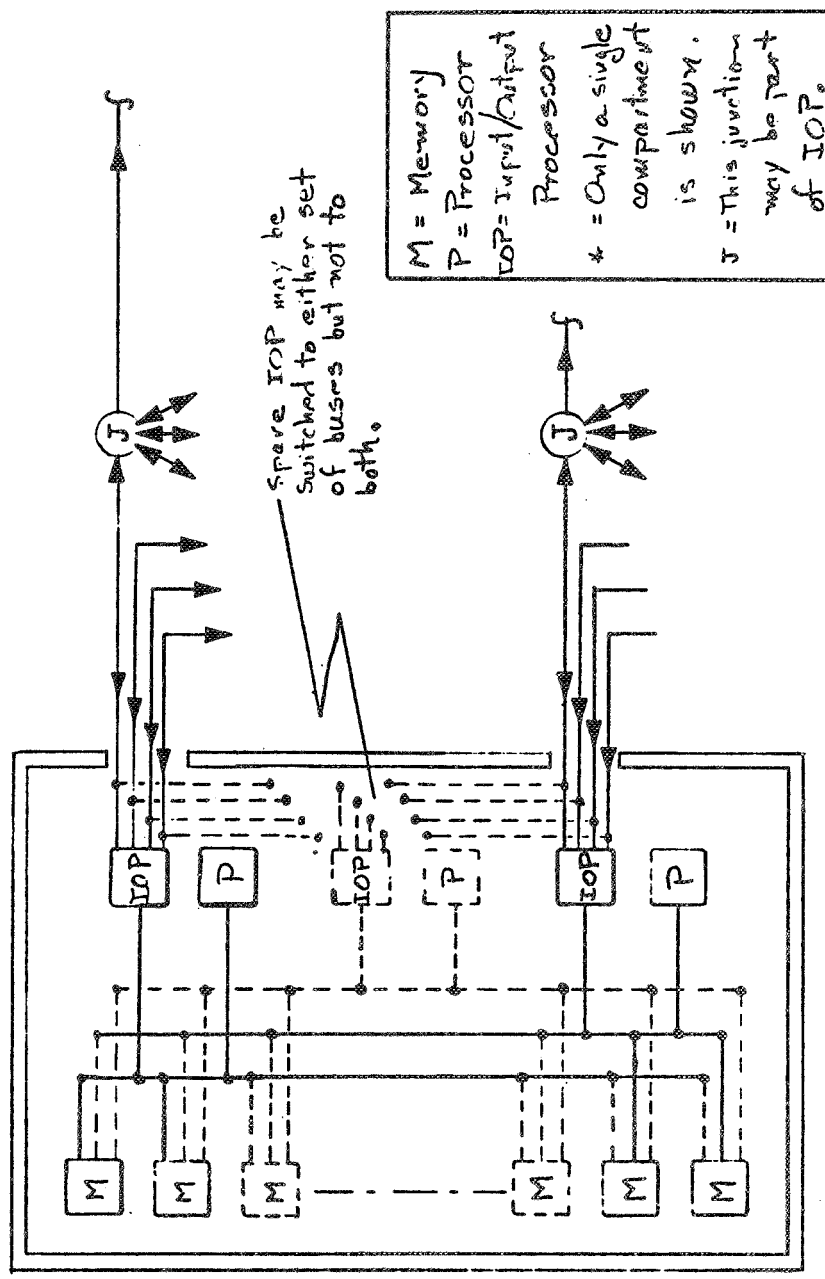


FIGURE 4-28. CONFIGURATION 3C WITH MODULAR MULTIPROCESSOR

4.6.3.4 (continued) - To do this safely would require some type of voting process. Either a majority of the computers would have to issue the commands or perhaps the logic to issue that command could be based upon the other computers supplying some information. Some hardware features might also be introduced. One feature in particular seems worthwhile. That is to have the BIT associated with a processor disable the capability of that processor to issue commands and to release all modules associated with that processor whenever a BIT detected failure occurs.

The fact that any memory module may be connected to any processor or IOP in the same compartment has a definite benefit over the multicomputer organizations. A single spare memory module can serve as backup for either of two "computers". Also, any good modules of a failed computer can be used as spare modules for the adjacent computer. For example, the non-modular multiprocessor cannot be reconfigured after the first failure. But should the adjacent computer subsequently fail, it could possibly reconfigure using the good modules of the previously failed computer in the same manner that the modular multicomputer uses spare modules.

In summary, the multiprocessor organization has virtually no affect on the operational programs since they will be operated in a multicomputer manner. Switching of modules is more complex but is offset by the advantage of providing a more flexible system in comparison to the multicomputer system. This added flexibility increases the complexity of the reconfiguration software because of the additional paths possible.

The self-test for the multiprocessor is almost identical to self-test for the multicomputer. Some additional testing will be required because of the additional "lock out" logic and other hardware features required.

The self-test and reconfiguration of the modular multiprocessor in system configuration 3C will be slightly more complex than configuration 2B since the self-test will have to test the voter/switch.

With the multiprocessor, one processor/memory can transmit on a second bus by switching to the adjacent IOP. In 3C any IOP can transmit on any one of the four busses as long as the corresponding voter/switch is operating. This ability to transmit data on the busses from the other compartment provides the only additional reconfiguration paths but these will be routed automatically by the voter/switch in most cases. The computers will have control of the voter switch to cause it to adapt after a failure or a reconfiguration and also for self-test purposes.

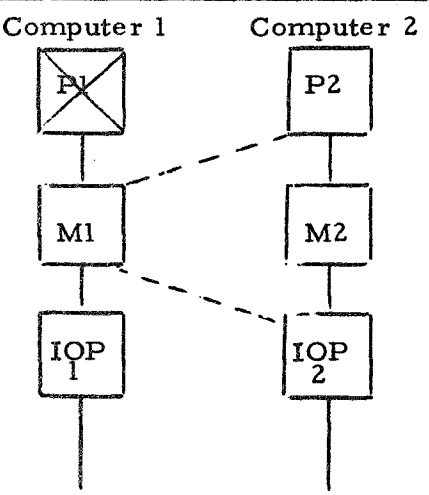
4.6.3.5 Non-Modular Multiprocessor - The non-modular multiprocessor can only be reconfigured after the adjacent computer has failed, and then only if the same module (memory, processor, IOP) did not fail in both. Reconfiguration in this case is relying upon two failed computers to reconfigure themselves into one good computer. To gain confidence that any reconfiguration can be accomplished, it seems reasonable that the

4.6.3.5 (continued) - reconfiguration paths be determined prior to the second failure within the compartment and the information relayed to the computers in the other compartments.

IOP failures, since they don't offset the self-test or reconfiguration ability of the processor/memory can be handled autonomously by a multiprocessor . If the first IOP fails, its own processor/memory and the adjacent processor/memory both are aware of it and are capable of reconfiguring. If the second failure is the other IOP, no reconfiguration is possible. If the second failure is either a processor or memory, there remains a second processor/memory combination capable of analyzing the situation and reconfiguring, assuming that agreement to reconfigure can be obtained from the other compartment.

Memory or processor failures frequently eliminate the processing capability of the "computer" and consequently remove it from partaking in the reconfiguration process. An example will help clarify this. Suppose we have the situation pictured below (Figure 4-29).

FIGURE 4-29 NON-MODULAR MULTIPROCESSOR RECONFIGURATION



Assume the first failure to be P1. As soon as the voter indicates this failure, "computer 2" analyzes the failure and isolates it to P1. It knows that if P2 fails no reconfiguration is possible. If either IOP fails, then P2/M2 can detect this and reconfigure without outside help. But if M2 fails, then P2 must be connected to M1. But a program in M2 cannot be relied upon to do this since it has failed. However, if after P1 failed, "computer 2" detected this and reported to the other compartment computers of the failure; then when the next failure occurs, the other compartment recognizes it and causes the one possible reconfiguration to take place. However, since a majority agreement is required

4.6.3.5 (continued) - to switch and since the other compartments may also have experienced a failure, to get a majority requires "computer 2" to vote on the switch. The switch would have to retain that information until the computer in the other compartment sends confirmation of the "computer 2" failure, at which time the switching will occur.

Another possible means of treating this situation is to provide a fixed program capability (a small read-only memory) in each processor for a minimal reconfiguration routine. This would essentially provide the processor some independence from the memory for reconfiguration. In the above example, once "computer 2" detected the failure in P1, it would notify the reconfiguration program and the other compartment. Now if M2 fails, the processor, P2, and at least one computer in the other compartment can vote on the reconfiguration path.

In summary, the non-modular multiprocessor cannot be reconfigured internally until after two failures have occurred in the same compartment, which means that as many as three computers may fail before reconfiguration is possible. Since two failures had to occur in the same compartment, reconfiguration becomes more complex because additional information is required from either the other compartment or another device, such as a fixed processor program, to confirm the reconfiguration path.

4.6.4 Other Considerations

In the above discussion it was always assumed that the basic operation was three computers processing redundant programs in parallel using the fourth for back-up and for processing non-critical programs. For functions of error sensitivity 1 or 2 (Para. 4.2.3.1.1) the computations may be single or double redundant. If a failure occurs in one of these but does not affect any triple redundant computations, then the function would be assumed by the other computer(s) involved in the triple redundant computations before calling on the fourth computer. So that this level of reconfiguration is essentially a function of the software rather than the hardware.

In every situation, except possibly for quadruple redundant computations, it is assumed that the computer system is directing the voter as to how many and on which lines the vote is to be based.

One may get the impression that the two modular organizations are more flexible than the non-modular organizations because the previous discussions always assumed spare modules were available. It is quite possible that spare modules will not be provided, but the modular organizations will still be the more flexible for the following reasons:

4.6.4 (continued)

The non-modular organizations as described (Section 4.5) have only a single 32K memory per processor. The modular organizations have several memories totaling 32K, such as two 16K modules. In the case of the multiprocessor organizations, if it is non-modular two memory failures in a compartment will preclude any reconfiguration. If it were modular with two 16K memories, then a single memory failure disables one "computer" but the remaining memory module is now available as a spare to the adjacent "computer". This advantage is not present in the multicomputer organization because the computers are electrically isolated. Modules of one computer can never serve as spares for the adjacent computer for automatic reconfiguration.

There is another possible advantage of the modular organizations. In the non-modular organizations, a single memory failure may disable all computing capability of that computer. In the modular organization the failure of one memory module does not necessarily completely disable the computer since it can operate out of one of the other memory modules, assuming that appropriate hardware controls are provided. This could prove quite useful for self-test after a failure. By programming self-test routines in each of the memory modules, it is possible to diagnose the original memory failure and also maintain surveillance of the remaining modules until being manually serviced.

4.6.5 Summary

The simplest operation and reconfiguration is using the non-modular multicomputer in a quadruple redundant fashion. This satisfies the FOOS criteria but provides little or no spare computing capability and no reconfiguration capability except for discarding failed computers.

The modular multicomputer and non-modular multiprocessor appear to be of about equal complexity with the multiprocessor leaning toward software to accomplish reconfiguration and the multicomputer toward hardware with its reconfiguration module.

If no spare modules are provided then the modular multicomputer is essentially identical to the non-modular multicomputer. The modular multiprocessor with no spares does have additional reconfiguration paths over the non-modular multiprocessor (Section 4.6.4).

The self-test required for diagnosis and isolation to a computer module is not significantly different in any system except for adding tests for additional hardware such as a voter/switch in the 3C configurations or switches required by the multiprocessors and modular multicomputer.

4.6.5 (continued)

The reconfiguration routines expand in size and complexity as the number of reconfiguration paths increases. Only the modular multicomputer with spares appears to require the hardware mechanized reconfiguration module.

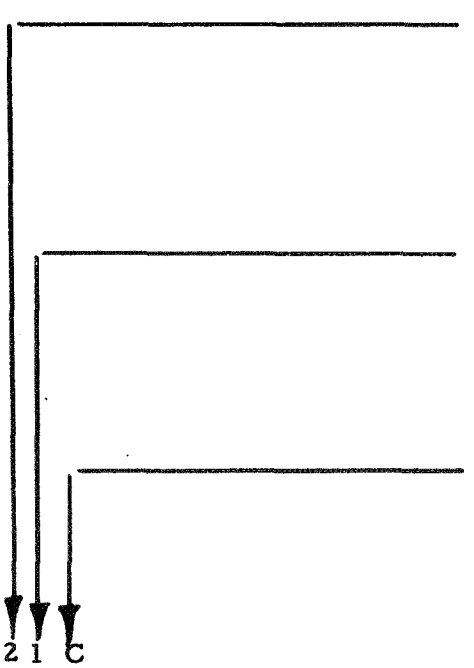
Assuming complexity of the reconfiguration software is not the deciding factor, then the modular multiprocessor appears to provide the maximum number of reconfiguration paths without the addition of spare modules.

4.7 QUANTITATIVE DATA FOR CANDIDATE COMPUTERS

4.7.1 Introduction

Section 4.5.2 defined the four internal computer organizations and Section 4.5.3 defined the architecture to be used to mechanize each organization. This section will present the quantitative data for the candidates which will be used in Section 5 to conduct the evaluation of the candidates. The estimates for the candidates were derived using current state-of-the-art technology techniques at Autonetics. This approach enabled the derivation of accurate data for the evaluation. Two technology approaches were used in these estimates as were defined in Section 4.5.4.

The numbering system used in describing all of the candidates will be given below:



Computer Organization:
1: Non-modular multicomputer
2: Modular multicomputer
3: Non-modular multiprocessor
4: Modular multiprocessor

System Concept:
1: Without VCS (concept 2B in Section 4.2)
2: With VCS (concept 3C in Section 4.2)

Technology:
C: Conventional (magnetic memory and conventional packaging)
A: Advanced (semiconductor memory and advanted packaging.)

4.7.2 Physical Characteristics

The physical characteristics will be given for the modules that comprise the organization (processor, memory, IOP, VCS) for each of the two technology approaches.

4.7.2.1 Processor Module - The architecture of the processor module is given in Section 4.5.3.1. Based on typical past designs of aerospace computers with this architecture, the physical characteristics were estimated. MOS/LSI technology was used for the logic in both technology approaches. A detailed description of the MOS/LSI devices used in the estimate is given in Appendix 5 of this volume. The estimates were derived for two types of processor modules; a basic module for the non-modular computer organizations (1 & 3) and a basic plus delta module for the modular computer organizations (2 & 4). The hardware estimates are given below in Table 4-2.

TABLE 4-2 PROCESSOR MODULE MECHANIZATION

Processor Module	Components
Basic	46 MOS/LSI 2 Hybrid thin film 30 Bipolar MSI IC 50 Discrete
Basic + Delta	46 MOS/LSI 2 Hybrid thin film 46 Bipolar MSI IC 82 Discrete

For the conventional technology approach the processor module is on one 8" x 12" multilayer (6) board with a 174 pin connector. Parts are mounted on both sides of the board individually with the exception of the hybrid thin film circuits.

The advanced technology approach for the processor module is on one 6" x 8: printed circuit board. The arithmetic and control functions use beam-leaded MOS/LSI devices mounted uncased (see Section 2.3) on a 2" x 2" ceramic substrate. Three of these substrates are used. The clock buffers and interface circuitry are contained on hybrid thin film circuits.

4.7.2.2 IOP Module - The architecture of the IOP was briefly described in Paragraph 4.5.3.3. In addition, the IOP contains one master serial channel receive/transmit), three receive-only serial channels, and a built-in test timer for self check purposes. As for the processor module, two types of IOP modules were estimated; a basic one for organizations 1 & 3 and a basic plus delta module for organizations 2 & 4. Based on previous designs for aerospace I/O processors, the following estimates were made as shown in Table 4-3.

4.7.2.2 (continued)

TABLE 4-3. IOP MODULE MECHANIZATION

IOP Module	Components
Basic	21 MOS/LSI 1 Hybrid thin film 36 Bipolar MSI IC 89 Discrete
Basic + Delta	21 MOS/LSI 1 Hybrid thin film 48 Bipolar MSI IC 121 Discrete

The packaging is the same as for the processor module, the conventional technology uses a 8" x 12" MLB and the advanced technology uses a 7" x 8" PCB.

4.7.2.3 Memory Module - Two basic types of memory modules were used: magnetic and semiconductor; each will be treated separately below.

4.7.2.3.1 Magnetic Memory Module

The conventional technology approach uses plated wire as the magnetic storage medium. Estimates are based on present prototypes developed using Autonetics' five mil plated wire. Two types of functional modules were estimated; a 32K x 32-bit word module and a 16K x 32 bit word module. The 16K module is used in the modular organizations while the 32K module is used in the non-modular organization. Further, a delta is required for each of the two types of modules to implement the memory modules for the multiprocessor organizations (Organizations 3 & 4). The memory operates with a 1μ second cycle time with read access time of 0.6μ seconds. A read/write ratio of four was used in deriving the power estimates. The non-modular multicomputer requires the simplest functions of the memory module; Table 4-4 contains the additional functions required of the other computer organizations.

4.7.2.3.1 (continued)

TABLE 4-4. MEMORY MODULE FUNCTIONS

Organization	Functions
Non-modular multicomputer	Basic 32K
Modular Multicomputer	16K, Module ID register, buffered bus for expansion.
Non-modular Multiprocessor	32K, Module ID register, priority control logic, 2 multi bus ports, lock out logic
Modular Multiprocessor	16K, Module ID register, priority control logic, 3 multi bus ports, lock out logic

The estimated physical characteristics for these modules are given in Table 4-5.

TABLE 4-5. MAGNETIC MEMORY MODULE PHYSICAL DATA

Module	Size (in^3)	Weight (lbs)	Power (Watts)
Basic 32K	1350	34.6	51
Basic 16K	790	23.7	39.2 (operating) 23.2 (stand-by)
Basic 32K + Delta (Org. 3)	1390	36.1	57.8
Basic 16K + Delta (Org. 4)	820	25.7	45.4 (operating) 29.4 (stand-by)

4.7.2.3.2 Semiconductor Memory Module - The advanced technology approach uses semiconductor memory with devices mounted uncased on 2" x 2" ceramic substrates. The substrates are then mounted as packages on printed circuit boards. The memory modules were mechanized with 1024-bit MNOS devices and 512-bit read/write MOS devices. It was assumed a 7:1 ratio of devices would be used, i.e., for a 16K module 14K of MNOS and 2K of MOS would be used. The same functions as listed in Table 4-4 apply in this case and will not be repeated here. The estimated physical characteristics for the memory modules are given in Table 4-6.

4.7.2.3.2 (continued)

TABLE 4-6. SEMICONDUCTOR MEMORY MODULE PHYSICAL DATA

Module	Size (in^3)	Weight (lbs)	Power (Watts)
Basic 32K	576	11.5	22.5
Basic 16K	320	6.4	16.2 (operating) 15.6 (stand-by)
Basic 32K + Delta (Org. 3)	576	12.0	29.3
Basic 16K + Delta (Org. 4)	320	6.65	22.4 (operating) 19.3 (stand-by)

4.7.2.4 VCS Module - The VCS module was functionally described in detail in Section 4.5.5. This module is used in the candidates that mechanize system concept 2. The estimate of physical characteristics for this module is given below:

Components

4 MOS/LSI
1 Hybrid Thin Film
3 Bipolar MSI IC
11 Discrete

These components are mounted on a two-sided printed circuit board. A 8" x 12" board is used for the conventional technology approach and a 7" x 8" board is used for the advanced technology approach.

4.7.2.5 Computer Module - This section presents the total estimates for each type of computer module based on the above data for each individual type of module in addition to estimates for power converter and clock functions. The data is presented in Table 4-7 for the 16 candidates (4 organizations x 2 system concepts x 2 technology approaches). In addition, a 17th candidate was added (4_{2c}*) as a result of feedback from the quantitative evaluation of the candidates presented in Section 5. This candidate is a variation of the modular multiprocessor in terms of packaging. Candidate 4_{2c}* assumes one physical package per compartment (two total per spacecraft), whereas all the other candidates assume two physical packages per compartment. The numbering system of the candidates is explained in Section 4.7.1.

4.7.2.6 Candidate Computer System Data - The computer modules listed below are combined in this section to form the set of physical data for each candidate computer system. The data given in Table 4-8 include cabling between the computers and, therefore, are not simply four times the individual module parameters.

TABLE 4-7. COMPUTER MODULE PHYSICAL DATA

Candidate	Size($Ft.^3$)	Weight (lbs)	Power(Watts)
$^{1}_{1C}$	1.21	67.0	103.5
$^{1}_{2C}$	1.28	71.1	106.0
$^{1}_{1A}$	0.526	25.8	65.1
$^{1}_{2A}$	0.56	28.0	67.8
$^{2}_{1C}$	1.56	88.88	123.0
$^{2}_{2C}$	1.63	92.98	125.5
$^{2}_{1A}$	0.58	28.56	81.7
$^{2}_{2A}$	0.62	30.1	83.2
$^{3}_{1C}$	1.24	69.3	112.6
$^{3}_{2C}$	1.31	73.4	115.1
$^{3}_{1A}$	0.58	26.3	74.5
$^{3}_{2A}$	0.625	27.8	77.0
$^{4}_{1C}$	1.61	104.2	140.0
$^{4}_{2C}$	1.68	108.3	142.5
$^{4}_{2C}$*	3.36	212.6	285.0
$^{4}_{1A}$	0.58	29.06	98.9
$^{4}_{2A}$	0.625	31.6	101.4

TABLE 4-8. CANDIDATE COMPUTER SYSTEM PHYSICAL DATA

Candidate	Size ($Ft.^3$)	Weight (lbs)	Power (Watts)
$^1_{1C}$	4.84	268	414
$^1_{1A}$	2.10	103	260
$^1_{2C}$	5.12	284	424
$^1_{2A}$	2.24	112	271
$^2_{1C}$	6.24	356	492
$^2_{1A}$	2.32	114	327
$^2_{2C}$	6.52	372	502
$^2_{2A}$	2.48	120	333
$^3_{1C}$	4.97	279	450
$^3_{1A}$	2.33	107	298
$^3_{2C}$	5.25	295	460
$^3_{2A}$	2.51	113	308
$^4_{1C}$	6.45	419	560
$^4_{1A}$	2.33	118	396
$^4_{2C}$	6.73	435	570
$^4_{2A}$	2.51	128	406

4.7.3 Reliability Data

4.7.3.1 Module Reliability - The modules discussed above were subject to a detailed reliability investigation in order to derive failure rates that could be used to generate computer system reliability for each of the candidates. Table 4-9 contains the failure rate data for the individual modules and the total for a computer module for each of the candidate computer systems.

Appendix 7 contains the details used in deriving the data in Table 4-9. The derivation is based on accumulated past history of similar components with suitable extrapolation to the space station time period of application.

4.7.3.2 Candidate Computer System Reliability - Reliability models were derived for each candidate computer system in order to determine the reliability of each candidate. The details of these derivations are given in Appendix 7. The failure rate data given in Table 4-9 was used with a mission time of six months to calculate the candidate reliability or probability of success. It should be noted here that the modular organization (2 and 4) candidates were assumed to contain no spare P, M or I/O modules in the reliability calculations. A summary of the candidate reliabilities is given in Table 4-10.

4.7.4 Miscellaneous Data

Size, weight, power, and reliability data for the candidates have been presented in the prior two sections, the remaining parameters for the candidates will be given below.

4.7.4.1 Cost - The cost data for the candidates is given in Table 4-11. This cost was developed from past experience with similar systems using the detailed parts lists developed for each candidate. The estimate includes both non-recurring and recurring costs for 20 systems. It is given as relative cost in Table 4-11 with candidate 1_{1A} being used as a reference base (cost = 1.0).

4.7.4.2 Growth Potential - The growth potential or expandability data is given in Table 4-12. Mint, Pint, and I/Oint are the initial amount of storage (in words)the initial number of processor modules, and the initial number of IOP modules respectively. Mmax, Pmax, and I/Omax are the maximum amount that these types of modules may be expanded to without any redesign or modification. Mmod is the module size or increment that can be added to expand the memory.

4.7.4.3 Software - Basically, each of the four organizations are operated identically, i.e., as a set of 4 multicomputers to solve the G&C task. However, there are some differences due to the presence of the VCS, reconfiguration paths, etc. Table 4-13 contains the total number of instructions

TABLE 4-9. PREDICTED RELIABILITY PER COMPUTER

(FAILURE / 10^6 HRS.)

Candidate Organization	*Package Convential (C) Advanced (A)	Processor	IOP	Memory	Power Converter	Chassis	VCS	Total (Comp Module)
1.1	C	2.7075	1.6566	12.3244	.3865	.1455	N/A	17.2205
1.1	A	2.2836	1.5899	43.1999	.3554	.1215	N/A	47.5503
1.2	C	2.7075	1.6566	12.3244	.3865	.1495	.3562	17.5807
1.2	A	2.2836	1.5899	43.1999	.3554	.1255	.3067	47.8610
2.1	C	2.8377	1.7868	15.1554	.3865	.1555	N/A	20.3219
2.1	A	2.4015	1.6971	48.5770	.3554	.1235	N/A	53.1545
2.2	C	2.8377	1.7868	15.1554	.3865	.1595	.3562	20.6821
2.2	A	2.4015	1.6971	48.5770	.3554	.1275	.3067	53.4652
3.1	C	2.7075	1.6566	13.0256	.3865	.1495	N/A	17.9257
3.1	A	2.2836	1.5899	43.8209	.3554	.1235	N/A	48.1733
3.2	C	2.7075	1.6566	13.0256	.3865	.1515	.3562	18.2839
3.2	A	2.2836	1.5899	43.8209	.3554	.1275	.3067	48.4840
4.1	C	2.8377	1.7868	17.1930	.3865	.1635	N/A	22.3695
4.1	A	2.4015	1.6971	50.4200	.3554	.1255	N/A	54.9995
4.2	C	2.8377	1.7868	17.1930	.3865	.1655	.3562	22.7277
4.2	A	2.4015	1.6971	50.4200	.3554	.1315	.3067	55.3122
4.2*								45.4554

TABLE 4-10. COMPUTER SYSTEM CANDIDATES RELIABILITY

CANDIDATES DESCRIPTION	CONVENTIONAL TECHNOLOGY		ADVANCED TECHNOLOGY	
	Without VCS	With VCS	Without VCS	With VCS
Non-modular Multicomputer	.999,968	.999,968	.998,094	.998,094
Modular - Multicomputer	.999,937	.999,937	.997,026	.997,025
Non-modular Multiprocessor	.999,986	.999,986	.998,598	.998,597
Modular Multiprocessor	.999,995,4	.999,995,3*	.999,958	.999,957

* Also applies to candidate 4_{2C}^{*}
* Also applies to candidate 4_{2C}^{*}

TABLE 4-11. CANDIDATE COST DATA

Candidate	Relative Cost
$^{1}_{1C}$	1.8
$^{1}_{1A}$	1.0
$^{1}_{2C}$	1.82
$^{1}_{2A}$	1.005
$^{2}_{1C}$	3.36
$^{2}_{1A}$	1.08
$^{2}_{2C}$	3.45
$^{2}_{2A}$	1.09
$^{3}_{1C}$	1.86
$^{3}_{1A}$	1.065
$^{3}_{2C}$	1.89
$^{3}_{2A}$	1.07
$^{4}_{1C}$	4.27
$^{4}_{1A}$	1.15
$^{4}_{2C}$	5.24
$^{4}_{2A}$	1.25

TABLE 4-12. GROWTH POTENTIAL

Candidate	M_{MAX}	M_{INT}	M_{MOD}	P_{MAX}	P_{INT}	IO_{MAX}	IO_{INT}
1_{2C}	128K	128K	0	4	4	4	4
1_{1C}	128K		0	4	4	4	4
1_{2A}	128K		0	4	4	4	4
1_{1A}	128K		0	4	4	4	4
2_{2C}	256K		16K	4	4	4	4
2_{1C}	256K		16K	4	4	4	4
2_{2A}	256K		16K	4	4	4	4
2_{1A}	256K		16K	4	4	4	4
3_{2C}	128K		0	4	4	4	4
3_{1C}	128K		0	4	4	4	4
3_{2A}	128K		0	4	4	4	4
3_{1A}	128K		0	4	4	4	4
4_{2C} & 4_{2C}*	384K		16K	6	4	6	4
4_{1C}	384K		16K	6	4	6	4
4_{2A}	384K		16K	6	4	6	4
4_{1A}	384K	128K	16K	6	4	6	4

4.7.4.3 (continued) - and data words that were estimated for each candidate. It should be noted that this does not represent the amount of 32-bit locations required since many of the instructions and data will be implemented with 1/2 word (16-bit) locations.

4.7.4.4 Interconnections - Table 4-13 also contains the listing of the number of connections (pins) required on each computer module. It accounts for all external interfaces (power, control panel, mass memory, busses, etc.)

TABLE 4-13. SOFTWARE AND INTERCONNECTION DATA

Candidate	Software (Instruction & Data Words)	Interconnection (Pins/Module)
$^{1}_{1C}$ - $^{1}_{1A}$	34,800	68
$^{1}_{2C}$ - $^{1}_{2A}$	35,100	70
$^{2}_{1C}$ - $^{2}_{1A}$	36.500	68
$^{2}_{2C}$ - $^{2}_{2A}$	36,800	70
$^{3}_{1C}$ - $^{3}_{1A}$	36,500	138
$^{3}_{2C}$ - $^{3}_{2A}$	36,800	140
$^{4}_{1C}$ - $^{4}_{1A}$	37,700	146
$^{4}_{2C}$ - $^{4}_{2A}$	38,000	148
$^{4}_{2C}$*	38,000	79

5.0 EVALUATION OF CANDIDATE COMPUTERS

5.1 INTRODUCTION

The objective of Task 5, Evaluation of Candidate Computers, is to select the best candidate computer system for further study and detailed definition. This effort can be broken into two distinct phases: (a) development of the evaluation model, and (b) actual evaluation of candidate computers by applying the model to candidates under consideration. This section of the report covers both of these phases. It defines the evaluation model and provides the rationale used during the development of the model. It also describes the evaluation process and summarizes the conclusions drawn from the evaluation.

Much of the rationale used during the development of the evaluation model was based upon information presented in the Work Statement, information obtained from the Space Division of NR, and the relative weighting factors provided by the NASA. The Work Statement and the Space Division information contributed primarily to defining the scope of the computer system and in turn the scope of the evaluation model, whereas the weighting factors were used as a basis for defining details of the evaluation model. Table 5-1 contains a list of the relative weighting factors as provided by the NASA. They are specified as both additive and multiplicative, and with respect to that candidate yielding the most beneficial value for the attribute under consideration. An attribute is defined to be a computer system characteristic that meets or exceeds the requirement for that characteristic.

Development of the evaluation model consisted of two basic activities: Selection of the evaluation method and definition of the computational details. The total effort was split about equally between the two activities. In selecting the method of evaluation, trade-offs were required in the following areas:

1. Basic Approach
2. Computational Technique
3. Data Normalization
4. Set Comparison Methods
5. Interpolation Schemes

Selection of the most appropriate direction in each of these areas was the substance of selecting the evaluation method. Similarly, definition of the computational details consisted of defining the following aspects of the selected evaluation method:

1. Attribute Definition
2. Method of Normalization
3. Quantitative Range of Attribute Value
4. Interpolation Scheme
5. Method of Combining Weighted Attributes

TABLE 5-1 - Weighting Factors Furnished By The NASA

Additive Attributes	Weighting Factors
Power	0.3
Weight	0.6
Volume	0.6
Cost (DDT&E and Flight Units)	0.7
Programming Ease	0.9
Reconfiguration Flexibility	0.6
Growth Potential	0.7
Probability of Success	1.0
Transient Immunity (Performance Wise)	0.9
Modularity (Functional, Physical, Replaceable)	0.7

Multiplicative Attributes	Weighting Factors	
"Ten Pin Rule" (10 → 50 Acceptable)	0.6	1.0
Subsystem/Management Clarity	0.8	1.0
Technology Criticality	0.5	1.0
Clarity of Approach (Intangible)	0.9	1.0

5.1 (Continued)

The definition of these items was not selective in nature but rather developmental.

Four basic computer organizations were selected for evaluation:

1. Non-modular multicomputer
2. Modular multicomputer
3. Non-modular multiprocessor
4. Modular multiprocessor

Two system concepts were also considered: one with a voter/comparator/switch at each computer I/O section, the other without the VCS. In addition, two different technologies were used to implement the candidate systems. This resulted in a total of 16 candidates to be evaluated.

A detailed description of the evaluation model is presented in Section 5.2 of this report. It defines the method of evaluation, defines the quantitative information needed to perform an evaluation, and discusses the interpretation of data obtained from an evaluation. Section 5.3 discusses the rationale used during the development of the evaluation model with emphasis on the selection of the evaluation method. Section 5.4 describes the evaluation process, provides quantitative evaluation data on the 16 candidate systems considered, and summarizes the conclusions drawn from the evaluation.

5.2 DESCRIPTION OF THE EVALUATION MODEL

The computer system candidate evaluation model provides a relative evaluation of the candidates. It employs analog computational techniques and a pseudo-normalization scheme for evaluating the attribute data. On the attribute level, a relative comparison of the candidates is based upon a set comparison method where the average of the set is defined to have an attribute value of zero prior to the pseudo-normalization. The remainder of the set receives an attribute value based upon deviations from the average and a linear interpolation scheme. All ten attributes are evaluated in the same manner.

Quantitatively, the total relative value of a computer system candidate is defined as a weighted linear combination of the computer system attribute values. More specifically,

$$\$ = K_1 K_2 K_3 K_4(k_1P + k_2W + k_3V + k_4C + k_5PE + k_6RF + k_7G + k_8R + k_9T + k_{10}Mod)$$

where,

$\$$ = Total Relative Value of Candidate

K_1 = Ten Pin Rule Weighting Factor

K_2 = Subsystem/Management Clarity Weighting Factor

K_3 = Technology Criticality Weighting Factor

K_4 = Clarity of Approach Weighting Factor

P = Power Attribute Value

W = Weight Attribute Value

V = Volume Attribute Value

C = Cost Attribute Value

PE = Programming Ease Attribute Value

RF = Reconfiguration Flexibility Attribute Value

G = Growth Potential Attribute Value

R = Reliability Attribute Value

T = Transient Immunity Attribute Value

Mod = Modularity Value

k_1 =	4.3	k_6 =	8.6
k_2 =	8.6	k_7 =	10.0
k_3 =	8.6	k_8 =	14.3
k_4 =	10.0	k_9 =	12.8
k_5 =	12.8	k_{10} =	10.0

5.2 (Continued)

The multiplicative weighting factors K_1 through K_4 can be more accurately defined as risk factors. As a result, their influence on the total relative value of a candidate is more significant than any other factor. The multiplicative weighting factors are variables, dependent upon the design of a candidate. Detailed definitions of these factors are presented later in this section. The factors k_1 through k_{10} are defined as additive weighting factors. They indicate the relative importance of the various computer attributes such as power, weight, volume, etc.

The method of computing the attribute values for a given set of candidates is as follows:

1. For each attribute, add the respective values of all candidates and divide by the number of candidates to obtain the average of the attributes. For example,

$$P_{AVE} = \frac{1}{n} \sum_{i=1}^{n} P_A$$

where,

P_A = the power attribute

P_{AVE} = the average of the power attributes

n = the number of candidates

By definition, the average of the attributes has an attribute value (intermediate value prior to normalization) of zero. From the example, the power attribute value for P_{AVE} equals zero.

2. Find the difference between each candidate's attribute and the average attribute to obtain the attribute deviation. For attributes where decreasing values are desired, subtract the individual attribute from the average. Where increasing values are desired, subtract the average for each attribute. Continuing with the example,

$(\Delta P_A)_i = P_{AVE} - (P_A)_i$ Decreasing values are desired for the power attribute.

$(G_A)_i = (G_A)_i - G_{AVE}$ Increasing values are desired for the growth attribute.

5.2 (Continued)

3. Computer intermediate attribute values for each candidate by dividing the deviation for each candidate by the average.

$$(P'')_i = \frac{(\Delta P_A)_i}{P_{AVE}}$$

Furthermore, limit the range of this intermediate value to between -1 and +1 by letting all attribute deviations that are more than double the attribute average have a value of +1. The range for the example is then

$$-1 \leq (P'')_i \leq +1$$

4. Modify each of the above intermediate values by adding 1 and dividing by 2. This in effect changes the range from "-1 to +1" to "0 to +1". The modified intermediate value in the example is

$$(P')_i = \frac{(P'')_i + 1}{2}$$

where,

$$0 \leq (P')_i \leq 1$$

5. Obtain the final attribute value by multiplying the modified intermediate values by a scale factor equivalent to the inverse of the maximum modified intermediate value. In the example,

$$(P)_i = \frac{+1}{(P'_{MAX})} (P')_i$$

where,

$$0 \leq (P')_i \leq 1$$

6. Substitute these values into the weighted linear combination equation.

Graphically, steps 1 through 3 can be depicted as shown in Figures 5-1 and 5-2. Figure 5-1 is the case where decreasing attributes such as power, weight and volume are desired; and Figure 5-2 is the case where increasing attributes are desired such as reliability, growth, and modularity.

5.2 (Continued)

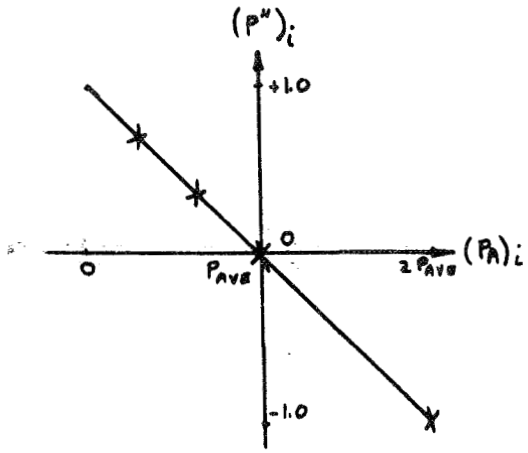


Figure 5-1. Decreasing Attributes Desired

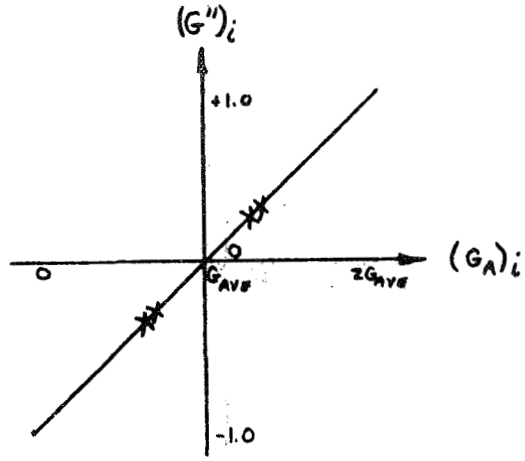


Figure 5-2. Increasing Attributes Desired

After applying step 4 of the above procedure, Figures 5-1 and 5-2 result into Figures 5-3 and 5-4, respectively.

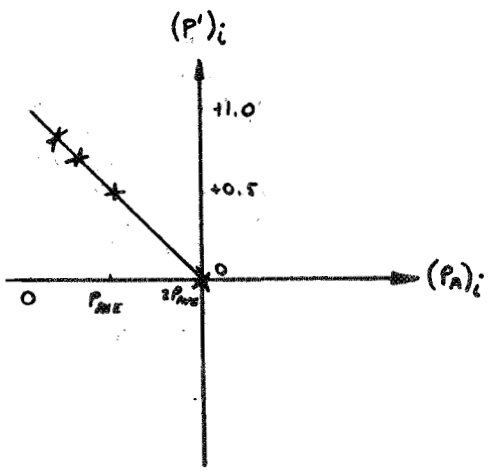


Figure 5-3. Intermediate Values

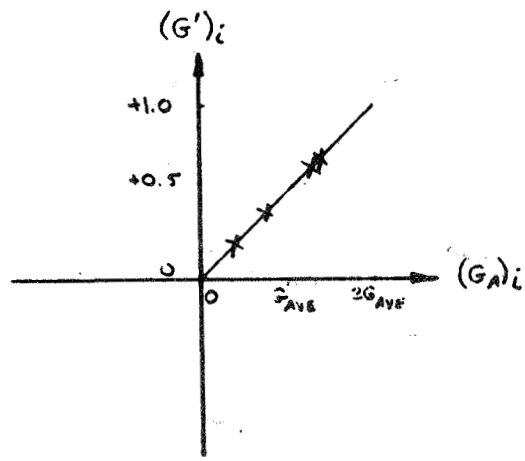


Figure 5-4. Intermediate Values

5.2 (Continued)

And finally, after applying step 5, Figures 5-3 and 5-4 result into Figures 5-5 and 5-6, respectively.

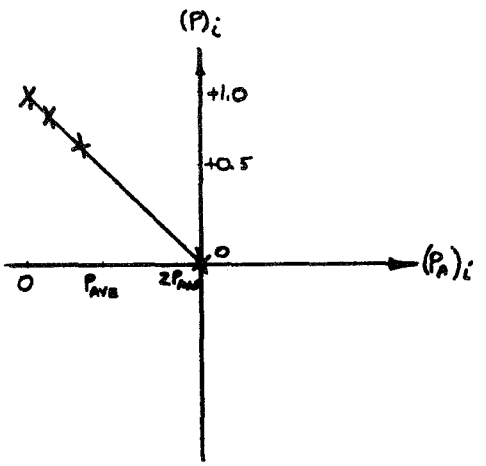


Figure 5-5. Final Attribute Values

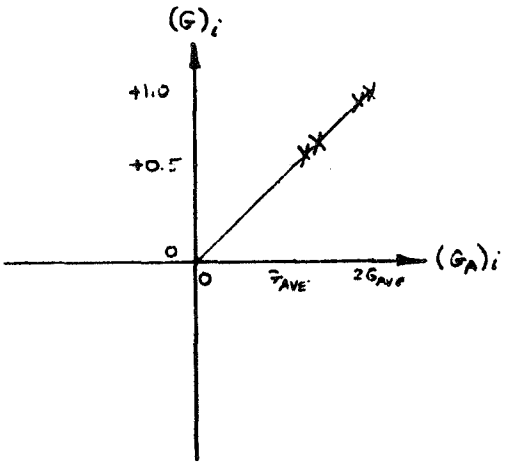


Figure 5-6. Final Attribute Values

From these illustrations, it can be seen that the final attribute values will have a range from 0 to 1, and that the best attribute will always have a value of 1.

The above method of computing the attribute values will be applied to all ten attributes.

Since the attribute values are highly dependent upon the attributes of each candidate, care must be taken to accurately determine each attribute. But even more important, the effort spent in determining the attributes should be split equally among the candidates; i.e., the confidence level in one candidate's attributes should equal that of another. The importance of this aspect lies in the fact that the average of the attributes is the basis for determining the attribute values; therefore, the accuracy of the evaluation can be easily perturbed by inadequately determining the attributes of only one candidate.

In the following paragraphs, a definition is provided for each of the ten attributes.

Power: The power attribute (P_A) is defined as the total power consumption of the candidate computer system that is simultaneously drawn from all redundant power busses during a typical GN&C maneuver in the manned

5.2 (Continued) orbital coast phase. It shall include all computer modules whether they be active, dormant or in a standby mode. It does not include preprocessor power or power required for voting logic that is remotely located from the centralized computer. The power attribute is expressed in terms of watts.

Decreasing values are desired for the power attribute.

Weight: The weight attribute (W_A) is defined as the sum total weight of all computer system modules, all inter-module cabling or bussing and the base chassis structure to which all modules are mounted. Spares are not under consideration (study ground rule). It does not include coldplate weight, forced air ducting and blower weight, interface cabling to external systems, pre-processor weight, control and display weight, and chassis structure external to the module and module base weight. The weight attribute is expressed in terms of pounds.

Decreasing values are desired for the weight attribute.

Volume: The volume attribute (V_A) is defined as the volume of the smallest rectangular parallelepiped that will fully enclose all modules of the computer system. Should the computer system consist of two or more main assemblies (each of which may contain several modules), then the volume would be the sum total of all the rectangular parallelepipeds. The volume does not include cabling to external systems but does include inter-module cabling. It will include all handles, connectors and protrusions. The volume attribute is expressed in terms of cubic feet.

Decreasing values are desired for the volume attribute.

Cost: The cost attribute (C_A) is defined as the sum total cost of the DDT&E effort, the production cost of ten (10) systems plus ten (10) complete spares, and the software cost. These costs cover only the centralized part of the computer system. They do not cover the preprocessors or data busses. Maintenance and GSE costs are not included. The cost attribute is expressed in terms of dollars.

Decreasing values are desired for the cost attribute.

Programming Ease: The programming ease attribute (PE_A) is presently defined as being inversely proportional to the total number of uniquely programmed instruction words required. It does not include any programs that are completely redundant where completely implies that the redundant routines need not be reprogrammed. Due to different computer architectures, a special modifier has been added to this attribute. This modifier normally has a value of one, but it may be less than one if a particular candidate is more difficult to program.

Mathematically,

$$PE_A = \frac{k_{pe}}{N}$$

where,

5.2 (Continued)

k_{pe} = architectural modifier

N = number of uniquely programmed instruction words

Increasing values are desired for the programming ease attribute.

Reconfiguration Flexibility: The reconfiguration flexibility attribute (RF_A) is defined to be a measurement of the number of different operational data flow modes of the computer system. The modes will involve only memory, processor, and input/output modules or submodules. They may be either automatically or manually commanded, but they must be electrically reconfigurable. Power supply and clock reconfiguration modes are excluded.

Increasing values are desired for the reconfiguration flexibility attribute.

Growth Potential: The growth potential attribute (G_A) is defined as that growth that is possible by physically adding memory modules, processor modules, and input/output modules. Hardware modification to existing modules is not considered. Additions to the computer system cannot degrade existing performance.

Mathematically,

$$GA = K_M G_M + K_P G_P + K_{IO} G_{IO}$$

where,

K_M = 10, K_p = 2, K_{10} = 5

G_M = Memory Growth

G_P = Processor Growth

G_{IO} = Input/Output Growth

Memory Growth is defined as

$$GM = \frac{M_{MAX} - M_{INT}}{M_{MAX} + M_{MOD}}$$

where,

M_{MAX} = Maximum number of full length memory words feasible under design

M_{INT} = Initial number of full length memory words in first operational system

M_{MOD} = Minimum number of full length memory words that can be modularly added

5.2 (Continued)

Processor growth is defined as

$$G_P = \frac{P_{MAX} - P_{INT}}{P_{MAX}}$$

where,

P_{MAX} = Maximum number of processor modules that are feasible under design

P_{INT} = Initial number of processor modules in first operational system

Input/Output growth is defined as

$$G_{IO} = \frac{IO_{MAX} - IO_{INT}}{IO_{MAX}}$$

where,

IO_{MAX} = Maximum number of input/output modules that are feasible under design

IO_{INT} = Initial number of input/output modules in first operational system

Increasing values are desired for the growth potential attribute.

Reliability: The reliability attribute (R_A) is defined as the probability of mission success for the computer system for a 180 day mission. The hardware under consideration includes only the computer system modules and inter-module cabling. It does not include preprocessors and external data busses. This attribute is in addition to the fail op, fail op, fail safe criteria.

Increasing values are desired for the reliability attribute.

Transient Immunity: The transient immunity attribute (T_A) is defined as the ability of the computer system to perform without error during an electrical transient or an intermittent failure. This attribute excludes EMI problems associated with the external data busses. It does include errors that occur during reconfiguration, however, Of primary concern is the outputting of incorrect information. No quantitative measure has been defined which can accurately define this attribute.

Since this attribute is not quantitatively defined, the attribute values will be estimated based upon the single point failure mode analysis. The best candidate will have an attribute value of 1.0, and the poorest will have an attribute value of 0.7.

5.2 (Continued)

Modularity: The modularity attribute (Mod_A) is defined as the total number of inflight replaceable modules which constitute a part of the computer system. They do not include any preprocessor modules.

Increasing values are desired for the modularity attribute.

In addition to computing the attribute values for each candidate, the four multiplicative weighting factors must also be determined. A definition of these factors and the method for determining their quantitative values are provided in the following paragraphs.

Ten Pin Rule Weighting Factor: The ten pin rule weighting factor (K_1) is a numerical value corresponding to the worth of a candidate with respect to the maximum number of electrical interconnect pins that exist on any one of the candidate's modules. It is with respect to those pins required for flight operation, and it does not include covered test pins. A design goal of ten (10) pins has been extablished, and an upper limit of fifty (50) pins is highly desired. The ten pin rule weighting factor for a given pin count is defined as follows:

Pin Count (X)	Weighting Factor (K_1)
$0 < X \leq 50$	$K_1 = 1$
$50 < X \leq 100$	$K_1 = -.002X + 1.1$
$100 < X \leq 200$	$K_1 = -.003X + 1.2$
$200 < X$	$K_1 = 0$

Graphically, these relationships are as shown in Figure 5-7.

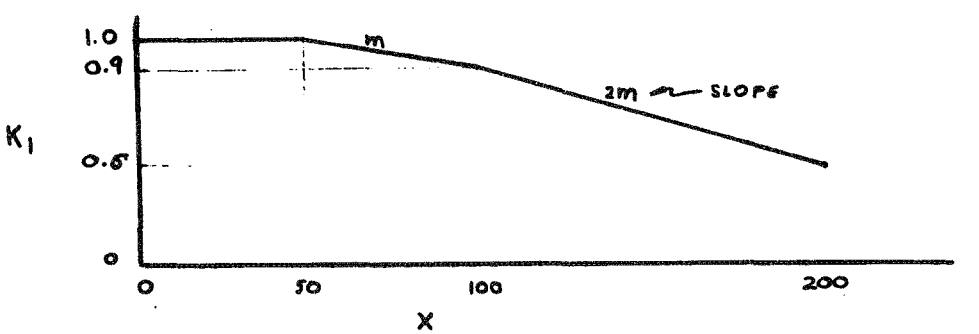


Figure 5-7. Ten Pin Rule Weighting Factor

5.2 (Continued)

Subsystem/Management Clarity: The subsystem/management clarity weighting factor (K_2) is a numerical value corresponding to the effectiveness of the working relationship between two or more companies or organizations. The effectiveness of the working relationship is proportional to and can be measured by the amount of different data that is exchanged via electrical interfaces. More specifically, the effectiveness is defined as the number of different pieces of information that enters or leaves the centralized computer. The weighting factor will then be determined from this information.

The subsystem/management weighting factor is determined in a manner similar to that for the attributes. An average for the interface data is computed taking all candidates into consideration. An intermediate weighting factor (K_2) for the average is defined to be 0.9 (not zero as in the case of the attributes). Deviations from the average are then computed. A candidate with twice the average has a value of 0.8, and a candidate with no data flow (an impossibility) has a value of 1.0. A scale factor equivalent to 1.0 / (K_2)MAX is then used to determine the final weighting factor.

The mechanics of determining this attribute are shown in Figures 5-8 and 5-9.

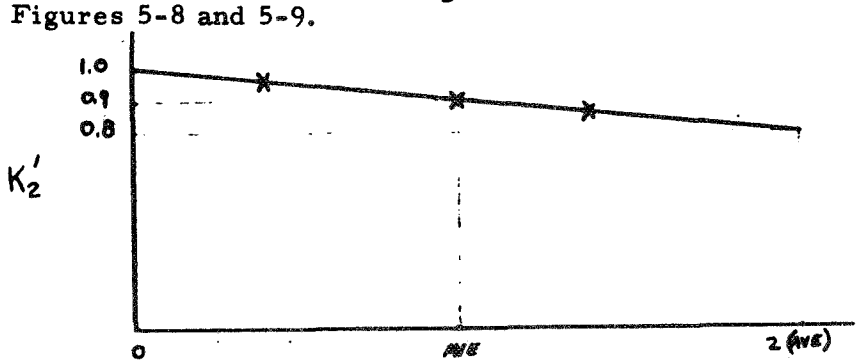
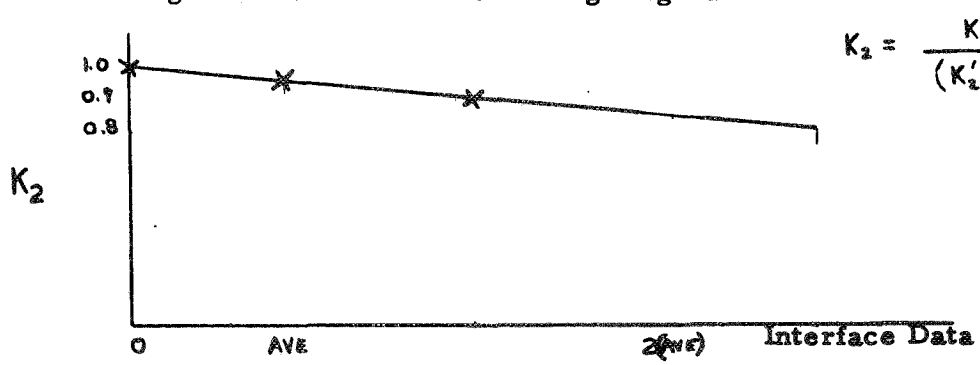


Figure 5-8. Intermediate Weighting Factors



$$K_2 = \frac{K_2'}{(K_2')_{MAX}}$$

Figure 5-9. Subsystem/Management Clarity Weighting Factor

5.2 (Continued)

Technology Criticality Weighting Factor: The technology criticality weighting factor (K_3) is a numerical value corresponding to the risk involved in using an advanced technology. The risk is usually in the form of jeopardizing the program by schedule slippage and/or developmental risk. Systems requiring large amounts of brute force engineering or systems that are extremely complex in their integration are not classified by being technological critical. However, new hardware or components requiring scientific breakthroughs, that have not been previously used, or where material discoveries are required, are critical.

The technology criticality weighting factors are specified by definition. Tables 5-2 and 5-3 define the weighting factors for the various circuit and memory technologies, respectively. The circuit technology weighting factors are a function of chip density, while the memory technology weighting factors are defined for various speeds. The weighting factor for a given candidate is defined as that value for the most critical technology used in the system.

Clarity of Approach: The clarity of approach weighting factor (K_4) is a numerical value which corresponds to the appeal of the technical approach upon the evaluation. The best approach has a value of 1.0 and the poorest approach a value of 0.9. For purposes of this study, it is constant of 1.0.

The general method for evaluating a set of candidates is as follows:

1. Determine the basic attributes for each candidate by analysis.

2. Determine the attribute values for each candidate from the methods specified in this section.

3. Determine the multiplicative weighting factors for each candidate.

4. On a candidate by candidate basis, substitute the attribute values into the total evaluation equation.

The end result from this process will result in a specific valuation for each candidate.

Table 5-2. Circuit Technology Weighting Factors

Complexity / Technology	ICs 20 Gates	MSI 20 - 50 Gates	Super MSI 50 - 100 Gates	LSI 100 - 500	Super LSI 500 Gates
Bipolar	1.0	1.0	0.9	0.7	0.5
4Ø PMOS	1.0	1.0	1.0	0.9	0.6
CMOS	0.9	0.9	0.8	0.6	0.5

Table 5-3. Memory Technology Weighting Factors

Cycle Time / Magnetic Memories	>2 us.	1 - 2 us.	0.5 - 1 us.	<0.5 us.
Plated Wire	1.0	1.0	1.0	.9
Core	1.0	1.0	.9	.7
Thin Film	.7	.7	.7	.7

5.2 (Continued)

The interpretation of the results is simple. The entire evaluation is on a percentage basis. For example, if a computer system was the best in all areas with respect to the other candidates and if the risk attributes were all ideal, that system would have a valuation of 100, $ = 100. A valuation of 100 is the highest possible valuation. More likely however, the best candidate will have a valuation in the range of 60 to 80. Relative comparisons are also simple. If the best candidate has a valuation of 75, and the second best candidate has a valuation of 60, then the best candidate is at least 25 percent better than all of the other candidates.

5.3 SELECTION OF EVALUATION METHOD

Prior to selecting a particular method of evaluation, a set of goals was established to insure that the evaluation model itself would be of value. If the evaluation model could not reliably evaluate the candidates, it would be of little use. Since the function of the evaluation model is to determine which computer candidate will receive further study and definition, its quality must be assured.

The most obvious goal was to make the evaluation model quantitative. A qualitative evaluation would be comparable to a sales pitch, and thus of little use. Two of the more important goals were to make the model accurate and highly objective. The accuracy goal is self-explanatory. However, the objectiveness goal is an aspect that is often overlooked in many evaluations. Evaluator subjectivity and corporate goals have a tendency to bias evaluations. Base of interpretation is a quality that makes the evaluation model acceptable; therefore, it was a goal. If the evaluator cannot easily understand the model and interpret the results, they obviously will not accept the conclusions. And finally, the mechanics of the model should be relatively simple; i.e., it should be easy to compute.

In summarizing the goals, the evaluation model was to be quantitative, accurate, highly objective, easy to interpret and easy to compute. Indeed, these goals have been met.

As a starting point, it was decided to define the value of the computer system as a weighted linear combination of the computer system attributes. Computer attributes can be thought of as a computer characteristics that meet or exceed the computer system requirements. It was assumed that all computer systems would meet or exceed the computer system requirements. Thus, all computer systems that are proposed as candidates are considered to be viable for the application. Numerous references are in agreement with this concept.

5.3.1 Options Available for Selection

The first major aspect in selecting the evaluation method was to determine which approach to use. Two main approaches were considered: A relative approach and an absolute approach. The relative approach provides a relative comparison of the candidates, whereas the absolute approach provides an evaluation with respect to the specific application.

5.3.1 (Continued)

In the relative approach, the evaluation is independent of the application. As such, it is not necessary to have detailed mission requirements during the study or at the time of the evaluation. The mathmatics are simple in the relative approach, because direct relationships between all candidates can be easily obtained. This implies that normalization and interpolation would be simplified, and that subjectiveness would tend to be remote. On the other hand, the worth or merits of this approach with respect to the specific application are not as high as they would be for an absolute comparison. Areas for candidate improvement with respect to the application are also more difficult to see.

The absolute comparison, if it is practical to use, is the best type of comparison. It evaluates the different candidates with respect to the application, and readily indicates specific areas for improvement for all candidates. However, normalization and interpolation are in general more complex. Thus, there is a tendency for subjectivity to be introduced into the evaluation. For this approach, the mission requirements and design goals must be known.

Within each approach there is a choice of computational techniques. Two that were considered during the study are an iterative discrete technique and a basic analog technique. The iterative discrete method requires iteration of the linear combination equation as many times as there are candidates. After each iteration one candidate is eliminated. The analog method requires only one iteration per evaluation.

The iterative discrete technique simplifies the mathematics and mechanics to almost nil. By eliminating normalization and interpolation, subjectivity is completely eliminated. However, the accuracy of this technique is unknown. Accuracy increases as the number of attributes increases, the number of candidates increases, and as the differences between the relative weighting factors becomes smaller. The output of the evaluation is an ordering of the candidates. Relative differences between candidates is not available.

The analog technique provides an output that is highly desirable and easily understandable. Relative comparisons of the evaluation results are automatically provided. This technique does not require a large number of attributes nor a large number of candidates to achieve accuracy. Furthermore, it does not restrict the weighting factors. This technique, however, is mathematically more complex. It requires computation of linear and possibly nonlinear functions. This in turn introduces subjectivity.

Independent of technique, a decision is required as to whether or not the variables should be normalized. Normalization of the variables provides for easier comparison of the candidates after the evaluation is complete, simplifies the computation of interpolation schemes, and reduces the subjectivity introduced by interpolation schemes. Normalization could distort the true worth of the variables, if it is not properly applied. Using un-normalized variables eliminates the possibility of distorting the worth of the variables and eliminates an added computational step from the evaluation. Un-normalized variables are more ideally suited to iterative discrete techniques, whereas normalization has more appeal for analog applications.

5.3.1 (Continued)

After selecting the approach, the computational technique, and whether or not to use normalization, a set comparison method was chosen. Four set comparison methods were considered during the study. The first method was defined such that the average attribute of the set equalled zero. Deviations from this average are then employed for computing the attribute value. The second method consisted of setting the required attribute for the set to zero. The third was defined such that the best attribute of the set equalled one, and the fourth was defined such that the worst attribute of the set equalled zero.

The average of the set equals zero set comparison method is highly objective and easy to use because of the simplified mathematics. The method gains in accuracy as the number of candidates increase. Due to the positive and negative character of the attribute deviations, a pseudo-normalization scheme is required to obtain the final attribute values. As a result, relative comparisons on the attribute level are more complex, and have a little less meaning than other set comparison methods. This method is more applicable to a relative approach than an absolute approach.

Using the required for the set equals zero method demands that the computer system requirements be known or estimated. If they are estimated, subjectivity is introduced. If the requirements are known, this is a good method. It is easy to compute, the results have more meaning with respect to the application, accuracy is good, and areas for improvement are readily seen. Although the meaning of the results is good, the interpretation and relative comparison, is only fair. Interpolation schemes are rather difficult to develop, and thus allow for more subjectivity. This method is more suited to the absolute approach.

The best of the set equals one set comparison method provides the best possible interpretation capability. It is easy to compute providing that the variables are normalized, and it is moderately objective for a wide range of variable values. Areas for improvement are fairly observable, and interpolation schemes are not too difficult to develop. Accuracy is affected by the number of candidates, and will increase as the number of candidates increases. Computer system requirements are not needed for this method. Objectivity diminishes if the number of candidates is reduced or if there are only small differences between the candidates. This method implies that the candidate with the best attribute of the set cannot be improved. Application of this method is best suited to a relative approach.

The worst of the set equals zero set comparison method is almost the inverse of the best of the set equals one. It is easy to compute and highly objective with accuracy being better near the poorer candidates. Normalization for this method is a must so that upper limits may be bounded. Interpretation of results is extremely difficult since no upper bounds are specified. For the same reason, interpolation schemes are difficult to develop. Due to poor accuracy in the area of the better candidates, this method is not too reliable.

5.3.1 (Continued)

For each set comparison method, many interpolation schemes can be defined. Those considered during the study were linear functions, parabolic and cubic functions, and estimation.

Linear interpolation is easy to compute and highly objective. Although interpretation is easy, it is less realistic than other schemes. This scheme provides the widest separation in attribute values over a given range of attributes resulting in the good interpretation. It is much better for an attribute with a short range, and much poorer for greater ranges.

Parabolic and cubic schemes are more difficult to compute than linear schemes, but they are much more realistic. They too are highly objective. They are more suited for evaluation over a wide range since they provide less discrimination in the area of interest.

Interpolation by estimation should be used only as a last resort. It is highly subjective and its realisticity depends upon the estimator. There is often difficulty in obtaining an estimate that is agreeable with all concerned. In some cases, this method may be the only method possible due to the inability to quantitatively define an attribute.

5.3.2 Selected Method

The relative approach was selected as the best approach for the evaluation model. This approach is simpler to compute, more objective, and not as demanding of the mission specifics as is the absolute approach. If all candidates are designed with the mission objective in mind, then the merits of the approach are quite meaningful.

Analog techniques were chosen as the mathematical foundation for the model. These techniques provided an accurate, highly desirable and easily interpretable output at the expense of more complex computations. The amount of subjectivity introduced by these techniques is highly dependent upon the details of the technique, and is felt to be minimal.

In conjunction with the above selection, the set comparison method that was selected is "the average of the set equals zero." This method is highly objective, is easy to compute, provides good interpretation of results. Furthermore, the system requirements need not be known. Assuming that at least four candidates will be evaluated, the accuracy of the method should be good.

With the selection of this set comparison method, a pseudo-normalization technique was employed. This resulted in obtaining the good interpretation of results associated with the "best of the set equals one" set comparison method. Thus, the good features of both set comparison methods were obtained.

A linear interpolation scheme was chosen for use with all attributes, except for the transient immunity attribute where estimation was employed. Linear interpolation is the more objective and easiest to compute of all schemes.

5.4 CANDIDATE SYSTEM EVALUATION

Sixteen candidate computer systems were defined based on four different computer organizations, two different systems concepts and two different technologies. A numbering system describing the candidates is defined in Section 4.7.1.

A detailed hardware description was prepared for each candidate, showing the number of LSI circuits, printed circuit boards and memory arrays. The hardware description also indicated size, weight and power estimates. Therefore, the hardware descriptions served as the primary sources of data for determining the attributes. The functional descriptions of the candidates presented in Section 4 of this report served as the basis for determining such attributes as the growth potential, transient immunity and modularity. Certain attributes such as the subsystem/management clarity, technology criticality and clarity of approach were based on judgmental factors to a certain extent.

The subsequent discussion provides assumptions and ground rules which were employed and are not apparent from the summary (Table 5-5) or from the basic definition of the evaluation model.

Power: The power attribute value (P) was based on the total power consumption of the computer system; i.e., four times the power of the individual computer power.

Weight: The weight attribute value (W) was based on four times the weight of the individual computers plus the following weight for inter-computer cabling: 1.6 pounds for non-modular multiprocessors and 2 pounds for modular multiprocessors.

Volume: The volume attribute value (V) was computer using the individual computer volume multipled by four plus the following intercomputer cabling volume: .008 cu. ft. for non-modular multiprocessor configuration and .009 cu. ft. for modular multiprocessor configuration.

Cost: Rough order of magnitude costs were developed by formula method based on past experience with similar computer systems and recent cost estimates for computers employing advanced technology. The estimate included both non-recurring and recurring costs for 20 systems. It was assumed that there would be no non-recurring costs for development of LSI circuits. The cost of flight software and software aids such as assemblies and simulators were included in the estimate.

Programming ease: The programming ease attribute value (PE) was based on the number of total instructions required to implement the G&C programs on the candidate computers. The attribute is inversely proportional to the number of instructions. Since the internal architecture of the candidates is almost identical, the architectual modifier k_{pe} was assigned a value of 1.0 for all candidate systems.

5.4 (Continued)

Reconfiguration Flexibility: The reconfiguration flexibility attribute value (RF) was based on the number of different data flow modes of the computer system. The probability of fault isolation and reconfiguration internal to the computers (arithmetic processor, memory module, I/O processor level) was also taken into account. It was assumed that module level reconfiguration is not always possible. Therefore, a probability of .5 was assigned to all internal reconfiguration paths, resulting in a more realistic measure of reconfiguration flexibility.

Growth Potential: The growth potential attribute value (G) was computed on the formulas presented in Section 5.2 of this report. Table 4-12 summarizes the modularity characteristics for the candidate systems.

Reliability: The use of mission probability of success (Ps) was found to result in a very minimal spread in attribute values even for widely disparate values of Ps. The same is true if probability of failure (P_F) is used. It was concluded that a new term, effective MTFB (Me) would be more meaningful and should be used in computing the reliability attribute values. The effective MTBF is defined as the mean time between failures which a non-redundant system would have in order to meet the same mission probability of success as the candidate system. The value of Me is calculated from Ps after Ps has been determined in the normal manner considering all redundancies and reconfiguration features of the candidate system. By definition,

$$M_e = \frac{t}{P_F}$$

where,

$$P_F = 1 - P_s$$

$$t = 4320 \text{ hrs. (180 days)}$$

This is good approximation for system where $M_e \gg t$. A detailed description of the reliability model used in the computation of P_s is presented in Appendix 7.

Transient Immunity: Due to the lack of good quantitative measure of transient immunity, the range of attribute values was defined from .7 to 1.0. An inspection of the candidate systems indicates that the most significant difference between the candidates, as far as transient immunity is concerned, is the presence or lack of the VCS at the data bus interface. Systems with VCS are able to detect transients as they occur and prevent them from being propagated through the data bus into the subsystems. Therefore, systems with VCS were assigned 1.0 and systems without VCS were given .7 for transient immunity attribute value.

Modularity: Modularity defined as the number of in-flight replaceable modules was determined to be the same as the number of computer packages; i.e., four for all configurations. Replacement of modules within each computer package was judged to be impractical due to limited on board capability for testing and repair. In this approach, sparing would be at the computer package level, thus providing rapid return to service capability.

5.4 (Continued)

Ten Pin Rule: The ten pin rule weighting factor (K_1) was obtained by a straight forward calculation based upon the number of pins required for external connection of each computer module.

Subsystem/Management Clarity: A value of 1.0 was assigned to all candidate systems since all candidates exchanged the same amount of data with other subsystems.

Technology Criticality: The technology criticality weighting factor (K_3) was determined from Tables 5-2 and 5-3. In case the circuit and memory technology factors were different for a given candidate system the lowest factor representing the most critical technology was chosen.

Clarity of Approach: The clarity of approach weighting factor (K_4) was set equal to 1.0 for all candidates, as specified in Section 5.2.

Table 5-4 shows the summary of the evaluation results.

The evaluation results point out several interesting facts:

Systems with VCS were always rated higher than the same systems without VCS. This is due primarily to the additional transient immunity provided by VCS outweighing the additional hardware and software needed. Therefore, systems without VCS were eliminated from further consideration.

Conventional technology offers substantial advantages over advanced technology. The difference is primarily caused by the higher risk of MOS technology. Also, it should be noted that the advanced technology systems were rated low in the areas of reliability. The relatively low reliability was due primarily to the higher failure rate of the MNOS semiconductor memory as compared to the plated wire memory. Therefore, systems mechanized with advanced technology were eliminated from further consideration.

Three candidates were identified as most promising: 1_{2c} (non-modular multicomputer), 2_{2c} (modular multicomputer), and 4_{2c} (modular multiprocessor). It became apparent at this point that the evaluation was very sensitive to the "ten pin rule." In particular, candidate 4_{2c} is severely degraded by this candidate. Consequently, 4_{2c} was considered in a different physical version using a total of 2 modules rather than 4 computer modules. This new version was labeled 4^*_{2c}. Evaluations were then conducted between candidates 1_{2c}, 2_{2c}, and 4_{2c}, and between 1_{2c}, 2_{2c}, and 4^*_{2c}.

The results of these evaluations are presented in Table 5-5. As indicated in this table, system 4^*_{2c} clearly has much higher total relative value than other competing candidates.

Table 5-4. Relative Value of Candidates

Sys	K_1	K_2	K_3	K_4	$\prod_{m=1}^{4} K_m$	$\sum_{n=1}^{16} K_n A$	$K_1 P$	$K_2 W$	$K_3 V$	$K_4 C$	$K_5 PE$	$K_6 RF$	$K_7 G$	$K_8 R$	$K_9 T$	K_{10} Mod	$
1_{1C}	.964	1.00	1.00	1.00	.964	56.29	3.09	4.54	4.68	5.55	12.80	1.72	0	4.95	8.96	10.00	53.3
1_{1A}	.964	1.00	0.50	1.00	.482	65.07	4.30	8.60	8.60	10.00	12.80	1.72	0	.09	8.96	10.00	32.0
1_{2C}	.960	1.00	1.00	1.00	.960	59.09	3.02	4.14	4.29	5.50	12.67	1.72	0	4.95	12.80	10.00	56.7
1_{2A}	.960	1.00	0.50	1.00	.480	68.22	4.22	8.39	8.38	9.95	12.67	1.72	0	.09	12.80	10.00	32.7
2_{1C}	.964	1.00	1.00	1.00	.964	52.92	2.48	2.36	2.68	2.98	12.19	1.72	7.01	2.54	8.96	10.00	51.0
2_{1A}	.964	1.00	0.50	1.00	.482	69.57	3.77	8.34	8.26	9.26	12.19	1.72	7.01	.06	8.96	10.00	33.5
2_{2C}	.960	1.00	1.00	1.00	.960	55.51	2.21	1.98	2.27	2.90	12.08	1.72	7.01	2.54	12.80	10.00	53.3
2_{2A}	.960	1.00	0.50	1.00	.480	72.82	3.72	8.18	8.05	9.20	12.08	1.72	7.01	.06	12.80	10.00	34.9
3_{1C}	.748	1.00	1.00	1.00	.748	64.38	2.81	4.27	4.50	5.39	12.19	5.16	0	11.10	8.96	10.00	48.1
3_{1A}	.748	1.00	0.50	1.00	.374	66.59	4.00	8.51	8.26	9.40	12.19	5.16	0	.11	8.96	10.00	24.9
3_{2C}	.740	1.00	1.00	1.00	.740	67.12	2.73	3.87	4.09	5.29	12.08	5.16	0	11.10	12.80	10.00	49.6
3_{2A}	.740	1.00	0.50	1.00	.370	69.80	3.92	8.35	8.00	9.38	12.08	5.16	0	.11	12.80	10.00	25.8
4_{1C}	.716	1.00	1.00	1.00	.716	71.10	1.94	.81	2.37	2.34	11.78	8.60	10.00	14.30	8.96	10.00	50.9
4_{1A}	.716	1.00	0.50	1.00	.358	81.59	3.23	8.24	8.26	8.67	11.78	8.60	10.00	3.85	8.96	10.00	29.2
4_{2C}	.708	1.00	1.00	1.00	.708	73.56	1.86	.42	1.97	1.91	11.70	8.60	10.00	14.30	12.80	10.00	52.1
4_{2A}	.708	1.00	0.50	1.00	.354	84.64	3.15	8.00	8.00	8.61	11.70	8.60	10.00	3.78	12.80	10.00	30.0

5.4 (Continued)

Candidate 4^{*}_{2c} was then chosen for further study and evaluation through software simulation. The recommended candidate was also redefined as the "restructurable multicomputer." The new definition is more descriptive since the system is used in a multicomputer mode of operation in order to satisfy the fail op, fail op, fail safe criterion. The multiprocessor data paths in the system are used for reconfiguration capability only.

Table 5-5. Additional Evaluation Between Competitive Candidates

CANDIDATE	$\prod K_m$	$\sum K_n$	$
1_{2c}	.964	71.46	68.7
2_{2c}	.960	68.92	66.2
4_{2c}	.708	90.01	63.6
1_{2c}	.964	71.46	68.7
2_{2c}	.960	68.88	66.2
4^{*}_{2c}	.942	84.74	79.7

6.0 I/O DATA BUS INVESTIGATION

6.1 INTRODUCTION

The I/O Bus Investigation is divided into five major sections. The first section covers the method of bus control and operation. The second section details the considerations involving the selection of a baseline data transmission technique for the data link. Also included at the end of this section are data cable considerations, clocking techniques and synchronization methods. A baseline or preferred approach is stipulated for each of these. Detailed design considerations for the data link are given in Appendix 9.

The third section covers in detail the error protection study performed and the results of that study. A selection of an error protection technique is made and an operational format for the bus specified to utilize this technique of error control. The fourth section discusses the impact on the bus system of the two candidate GN&C system configurations reported in Section 4.

Finally a summary of the preferred baseline mechanization of the I/O data bus is presented. This section brings together the salient features of the baseline mechanization and summarizes the overall report. Detailed operational sequences of computer to LP and LP to computer communications are not specified. These details are configuration dependent, and are detailed in the report section on the IOP and the local processor. The overall operation of the bus system is highly flexible and configuration independent, and it is this level that has been documented.

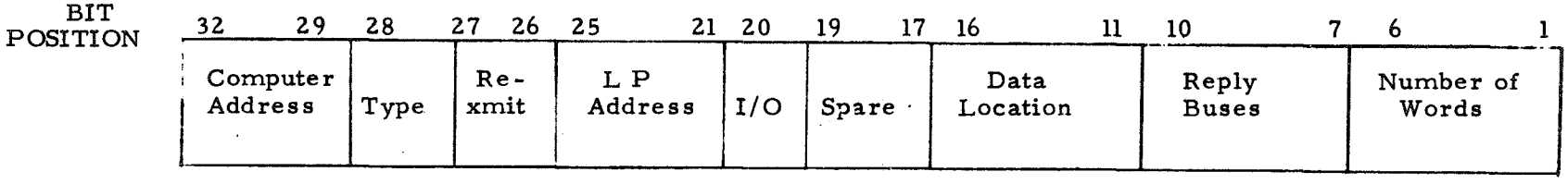
6.2 BASELINE BUS CONTROL

All communication on the data bus will be under computer control. All bus lines go to all computers indirectly through the VCS structure allowing each computer to monitor the operation and data on each of the four bus lines. Each bus link is dedicated to a computer, and its IOP has control over this bus. The IOP will initiate all bus communication.

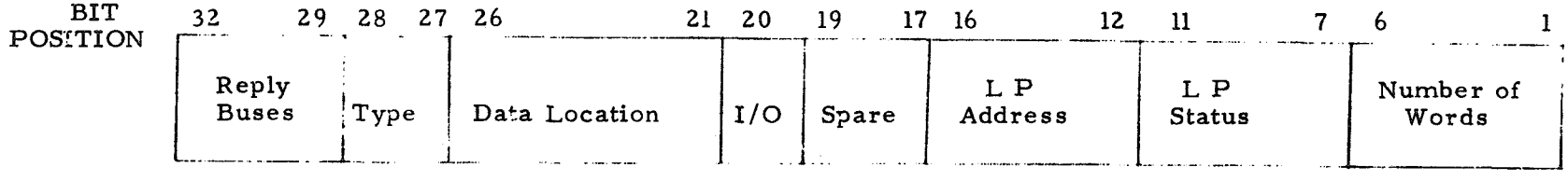
The bus operation can be divided into three major categories. These are: receive computer data, transmit data to computer, and other operational commands. During the receive computer data operation, each LP addressed will accept data from the computer for its associated subsystem. Under transmit data to computer operation, the addressed LP will transmit selected data from its subsystem on the appropriate bus. Command operation will be used to transfer commands from the computer to the LP for higher level control of the LP/Subsystem.

The bus operation will be defined as request-acknowledge communication. This means that for any request there will always be an acknowledgement from the LP. Only computers can make requests. The LP will be allowed to ask for a request in its acknowledgement signal.

All request messages will have an identical format, as will all acknowledgement messages. The coded request messages will be the primary methods of initiating any bus operation. Two bus control request words will be used for all three categories of operation. These words will be constructed as shown in Figure 6-1. They are shown as a single 32 bit computer word at the IOP which becomes two 16 bit words on the data bus and in the LP. Appended to the start ofthese words is a three



COMPUTER TO LOCAL PROCESSOR CONTROL WORD



ACKNOWLEDGE CONTROL WORD

FIGURE 6-1. CONTROL WORD FORMATS

bit sync code discussed later. These words will precede all data messages occurring each time the IOP initiates communication with a subsystem.

6.2.1 Computer to Local Processor Control Words

These control words are used when data is to be sent to or requested from a local processor. The fields of the control words are defined as follows:

Field	Bits	Definition
Computer Address	4	Indicates which computers are to receive the control word. Any combination of computers may be addressed by setting the appropriate bits to ONE.
Type	2	Identifies control word type.
Retransmit	1	ONE: Message is to be sent again if an error is detected in the transmission. ZERO: Message is to be sent once.
LP Address	5	Identifies the local processor being accessed by the computer.
I/O	1	ONE: Data are to be input to the computer. ZERO: Data are to be output from the computer.
Data Location	6	Identifies the LP memory address which contains the starting LP memory data location.
Reply Buses	4	Identifies the buses over which data are to be transmitted from the local processor to the computer.
Number of Words	6	Indicates the number of data words in the message.

The LP address field will have a distinct code word for each addressable LP, up to a total of 32. This allows communication with any single LP. Any subsystem with more than one Local Processor can have a separate or identical address for the additional unit.

The spare field bits can be used to delineate the operation of the LP. The basic transmit or receive data operation is the I/O bit. Other possible control items will be power-on-off to subsystem; special transmit routines for self-test, such as ADC calibration checks; different modes of data entry or transmission, such as retransmit all received data; request for status register contents; etc.

6.2.2 Acknowledge Control Words

These control words are generated by the LP and sent to the computer. The control words are sent to acknowledge the receipt of data by the LP or to start the transmission of data by the LP.

Acknowledgement messages will always be transmitted after receipt of a request. These messages will also have a common format utilized by all LP's. This format is also shown in Figure 6-1. An acknowledge is always preceded by a three bit sync code. The sync field will be used for synchronization of received messages at the IOP.

The requested (addressed) LP will respond with its hard wired address in the address field. This is a check on which LP is transmitting the acknowledgement message. The control fields will be identical to the ones received as another check on operation of the LP as well as on proper receipt of commands when no other action is requested.

An LP status field will be defined which will give gross indicates of the current operation of the LP. These flags will indicate such things as:

1) Parity error in request word (or other error detection results).

2) Parity error in received data (or other error detection results).

3) Result of power control command and other commands.

4) Gross LP status based on self-test (BITE).

5) Gross subsystem status based on self-test (BITE).

6) LP request for complete data dump to computer.

Commands performed by the LP will be such that a parallel real time indication of the presence or absence of the command signal will always be available for inclusion in the acknowledge word or on a later request for status by the IOP. The other fields of the acknowledge word are as defined in 6.2.1.

6.2.3 LP Operation

The LP operation for the three functional categories follows set patterns. These are listed in Table 6-1.

Table 6-1. LP Operation

OPERATIONAL MODES
Data to LP From Computer
Receive Request
Receive Data
Transmit Acknowledge
Data From LP To Computer
Receive Request
Transmit Acknowledge
Transmit Data
Operational Command For LP Or Subsystem
Receive Request
Perform Required Command Switching (Parallel Operation)
Transmit Acknowledge

Bus switching is done at the individual LP's to reply to the IOP's. This information is sent to the LP in the second control word in the reply buses field. The LP reply can be transmitted on one, two, three or all bus lines back to the central computer complex.

Reconfiguration information can also be provided in the request and acknowledge words as needed. This is one type of higher level LP control or status information. One of the data words available to the computer on request (for data) is a detailed BITE status word(s). This can be requested on an "as needed" basis or any other frequency that might be desired. It will always be transmitted to the computer when a full data "dump" is performed, since it is handled in the same manner as any other data word.

Appropriate no-data timing intervals will be included in the overall communication scheme for "guard"bands between words or messages. This principally occurs after the computer finishes transmitting and before the LP can respond. They are necessary to adjust for timing delays and skew between different users of the bus. The data clocking technique will be such that all data is disassembled under control of the clock used to originally assemble it.

Detailed operational sequences at the LP are described in the Local Processor Section. The IOP/VCS operation is described in its section and not repeated here.

6.3 I/O BUS LINK

The job of the data link is to provide at its output a replica of the signal applied to its input. The data link for this discussion includes only that equipment required to transmit a signal from one one point to another. The link will be time shared between many users.

Reliable data transmission is accomplished by the use of error detecting or correcting codes and by guaranteeing a signal-to-noise ratio in the transmission link adequate to produce acceptably low error rates. This section of the report discusses and evaluates various modulation-demodulation methods to achieve this end.

The first consideration for the data link shall be reliability. In evaluating transmission techniques suitable for the data link, the following design rules were considered:

1) Design for high reliability;

2) Low error rates compatible with the system;

3) Minimum susceptibility to noise;

4) Minimal noise emitted by the data link;

5) Redundant operation;

6) Minimum components;

7) Design for low weight, size and cost and power;

8) Ease of maintainability with no operational adjustments.

The general philosophy for the design of the data link must encompass all the above ground rules with special emphasis on reliability. Stress will be placed upon minimizing the effect of noise as related to reliability. Selection of the best transmission technique for low error rate, and employment of low powered microcircuits, works hand-in-hand with a system that operates with small signal levels. All of the above will be used to reduce the level and effects of susceptible and emitted interference. It should be noted that the degree of noise rejection required cannot be determined without information about the levels and spectra of noise in the spacecraft.

6.3.1 Signal and Noise Considerations

The probability of bit error of a system is a function of the signal-to-noise ratio, the noise bandwidth and the data rate. The distribution of noise in the bandwidth of the communication channel is unknown. In many cases, it is assumed Gaussian, but for aircraft this assumption is not generally valid. It is assumed that the spacecraft conducted or common mode noise is not Gaussian either, and probably follows an 1/f characteristic similar to aircraft as in Figure 6-2.

The data transmission link has two conflicting requirements, low probability of error and low interference with other signals. The probability of error is inversely related to received signal power. This is true except at very low probability of error. However, increasing signal power increases generation of interference.

The energy per bit, E, is inversely proportional to frequency of data for a given signal power level. Thus, the signal power must be increased in proportion to the data rate to maintain a constant energy per bit. In digital data transmission, a certain amount of energy is required to make a decision as to whether the symbol representing a 1 or a 0 was received. The transmitted power must equal the product of the energy per bit and the bit rate plus the channel losses. The reliability of the decision as to whether a 1 or 0 was received is known to increase with the amount of signal energy that can be integrated during a bit time. With a given bit time and signal power, a data waveform or code symbol should be selected that will deliver the maximum energy to the receiver. The maximum likelihood of a correct decision occurs when the energy of the 1 symbol is equal to the energy of the 0 symbol. The symbols should have equal probability of detection.

The approach taken for this study is to improve the data transmission channel by reducing its susceptibility to noise. With reduced noise in the channel a good signal to noise ratio can be maintained with low signal power. This also aids in reducing the noise interference emitted by the data link. In considering external noise affecting the data transmission link, it is clear that noise can only enter the channel by conduction or radiation.

Conducted noise is caused by power supplies, motors, relays and other electrical devices which use a relatively large amount of power. These devices when operated can cause transient changes in the ground system currents and cause voltage differences in the ground system. The transient currents have broad spectral characteristics which make it nearly impossible to maintain equipotential grounds. The ground current noises are coupled from one ground system to another by common impedances between ground points. When a data channel is connected between two different nodes of the ground system, ground noise appears across the ends of the channel. This voltage is termed conducted or common mode noise and is the worst offender because of its high levels and unpredictable nature.

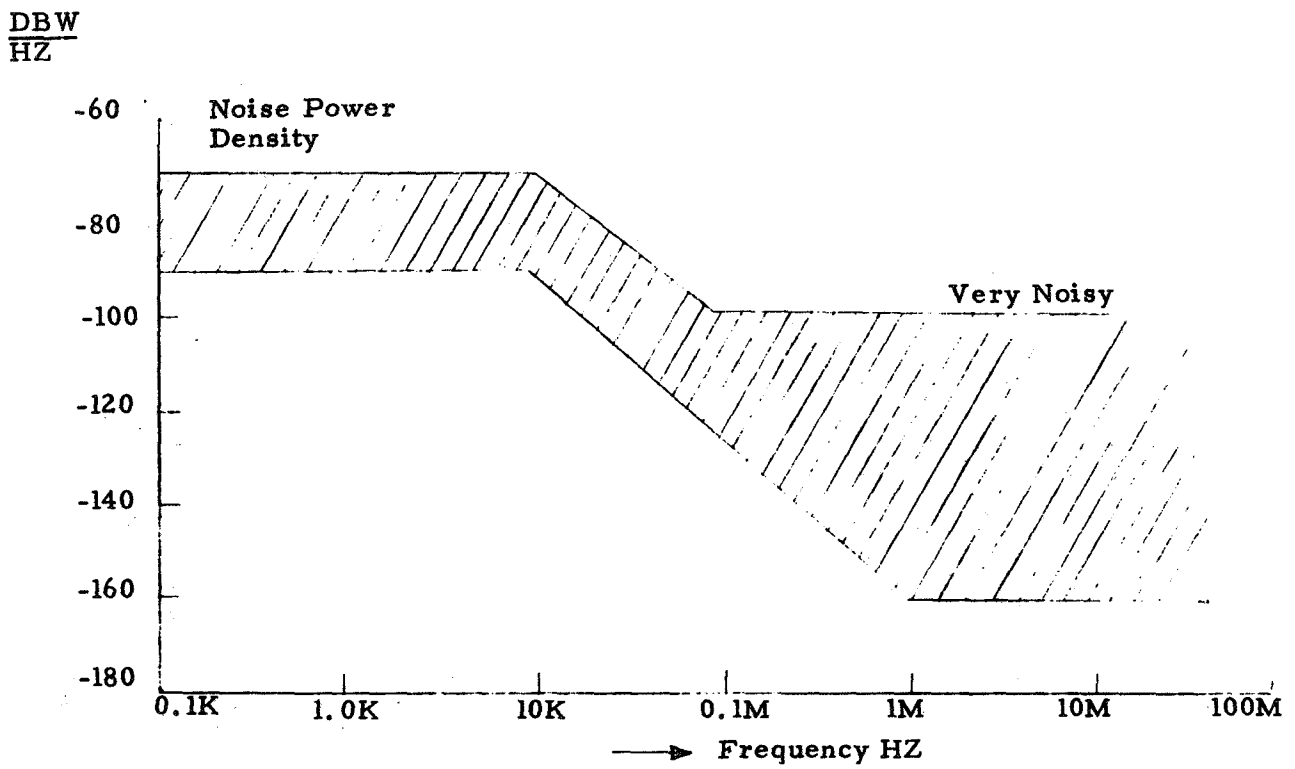


FIGURE 6-2. ASSUMED RANGE OF NOISE SPECTRUM

Noise coupled by capacitance is also present. Interference of this type is usually caused by high frequency, high voltage elements in close proximity to the data link.

Electromagnetic radiation interference usually enters a channel by field induction linking the closed loop formed by the signal path and the return path which form a data channel. The noise voltage induced by the loop antenna effect is directly proportional to the loop area.

The radio frequency interference emanating from a data transmission link must be kept low to prevent excessive interference with the operation of other systems. Analogous to the data link noise susceptibility problem, RFI generated internal to the data link is transferred from its source to other systems by either conduction or radiation or both. The radiated part of the RFI emitted by the data link is proportional to the loop area as well as the current in the channel.

The data link can therefore act as an RFI receiver or transmitter using the same mechanisms of conduction and/or radiation. In general, a cable that does not produce radiated or conducted RFI will not be susceptible to radiated or conducted RFI. In a practical sense, the radiated and/or conducted RFI can be reduced to a level which is no longer troublesome. Methods and hardware for accomplishing this are considered in the following paragraphs.

A method of rejection of common mode or conduction noise is to balance and isolate the data link from the terminal equipment grounds. Well balanced transformers and transmission lines provide several features that help improve the noise problem.

Current that is coupled through the transformer will be balanced, and transmitted equally. If the transmission path is balanced also, the noise will be cancelled and have no observable effect on the data signals at the receiver. Magnetic coupling of ground conduction noise, through the transformers, is also balanced because of the center tapped windings at the transmit and receive terminals. Currents induced by RFI radiation linking the loop also have no effect on the received data signal. From the preceding analysis, a well balanced isolated transmission system having small loop area is most advantageous.

External shielding without balance aids in the rejection of radiated and capacitive coupled interference. The degree of rejection, however, will prove less than that obtained by a well balanced system. Noise may be rejected by use of direct coupled lines with differential receivers which are biased to reject common mode and any other signals below a certain threshold. This type of noise rejection requires relatively high data levels.

There are other means by which the susceptibility of the data signal to the noise in the data link can be reduced. The base band spectra of the data signal can be separated from the noise spectra by shifting the frequencies of the information above the high power noise frequencies.

Pulse amplitude, rise time, and duration govern the level of interference emitted by a pulse. Increasing the energy per bit to transmit error free data increases the emitted interference directly. Thus, reducing the noise environment a signal will be subjected to, makes possible a reduction in the noise emitted by the signal.

Determining the most desirable data bandwidth and signal level with respect to hardware and noise is a complex trade-off. Some of the possible methods for implementing the data transmission link appear in the following section of this report.

6.3.2 Types of Data Transmission

There are several methods by which the data transmission may be accomplished. Three general methods of applying and recovering data signals are considered here.

For the following discussions, some definitions are applicable: The NRZ-L*, PCM (pulse code modulated) code from the multiplexer will be at a one megabit second rate. Bit clock will be supplied by the data link, both at transmit and receive terminals. Word or frame synchronizing will be accomplished in the multiplexer. At the receiver, NRZ-L, PCM code waveforms, a replica of the remote input, will be supplied to the multiplexer. Figure 6-3 is an illustration of various data encoding techniques.

6.3.2.1 NRZ Data Transmission System (Method A). An initial consideration leads to the possibility of transmitting the basic NRZ-L data without modification (encoding). Essentially, the NRZ-L data is passed through a low pass filter and applied to the line via an amplifier or suitable line driver. A crystal clock is well filtered and applied to the line also. The adding circuits for the filtered NRZ-L and clock are linear, and no intermodulation occurs. The filtering is accomplished by RC networks which will result in less complexity and weight. The transmission line will be well characterized on both transmit and receive ends to keep RFI at a minimum. The data power level on the transmission line will be reduced to produce low RFI emission and still allow low error rate.

The clock is a line spectrum signal, whose frequency is located well above the data transmission band, where it is easily separated at the receive terminal. The band limited data causes low interference at clock frequency. Since clock frequency is a multiple of data rate, spectrum energy of data will be low at clock frequency and data interference with clock will therefore be low. The time constant of the clock filters will be chosen for the smallest allowable time delay commensurate with system operation.

The advantage of this system is equipment simplicity because no encoding or decoding is necessary.

The disadvantages are:

1. The necessity of transmission to zero frequency, and in the low frequency region where the assumed noise is at its greatest. This will require the use of balanced transmission cables for signal to noise improvement and also necessitate DC coupling.

2. Bit sync must be transmitted separately since there may be a lack of data transitions over many bit periods. Phase lock synchronizers which derive bit timing from basic data are not attractive for operation under these conditions because of long pull in times and probable loss of data.

*NRZ-L (Non-return to zero-level) is data in which the level of the signal indicates the presence of absence of a "1".

FIGURE 6-3. ENCODING/MODULATION TECHNIQUES

6.3.2.2 Modem, Bi-phase Level (Method B). NRZ-L data is encoded to Bi-phase Level (Manchester) in which a transition occurs every bit time, in one direction for a "one" and the opposite direction for a "zero". A balanced transmission line is used with transformer coupling (AC coupled) and resistor isolation. A band pass filter is used at the receiver to reduce the noise bandwidth of the system. The filtering can be accomplished by either LC or RC networks. The transmission line will be well balanced at both ends to keep RFI at a minimum. The data power level on the line will be reduced to produce low RFI emission and still allow a low error rate.

The receive terminal decodes the Manchester to NRZ-L which will be a replica of the transmit input. To allow this, the receive clock must be an accurate reconstruction of the clock at the transmitter.

Important requirements for this clock are the preservation of the frequency and the phase of the transmit clock. Since the Manchester code contains a transition for every bit, and is synchronized by the transmitter clock, the code transitions contain sufficient information to reconstruct a proper receive clock. At the receiver input samples of the code transitions are made in the signal conditioner. These are phase compared with the output from the receiver voltage controlled crystal oscillator in a phase detector. The detected output closes the oscillator control loop. Loop parameters are selected to allow good frequency pull-in and close phase-lock.

The advantages of this method are:

1. Transmission system does not require response to DC and can be band limited, thereby reducing the interference susceptibility and emission.

2. Receive clock can be reconstructed from received data.

3. The transmission band is located in a lower noise band than that required to transmit the basic NRZ-L data.

The disadvantages are:

1. It requires complex means to derive received clock.

2. The circuit complexity is greater than that required to transmit only basic NRZ-L data.

Two other techniques are available for clocking and bit synchronization using this method of data transmission. In one, clock information is sent directly over the transmission path. The prime frequency of the clock is generated well above the pass-band required by the encoded data. The sine wave thus generated occupies a line spectrum which can be easily filtered from the data at the receive terminal. The actual clock at either terminal is the line oscillator frequency divided by eight.

Over and above the advantages of the previous system is the feature of reducing the complexity of the phase-locked loop to dividers at each terminal.

The main disadvantage of this technique is the jitter that prevails in the clock due to dividing. This may be made acceptable.

The second technique uses a clock at each transmitter and receiver and no clock on the line. This clock frequency is also approximately eight times the data frequency, and the same type of circuitry is used to derive the bit timing from the Manchester data using a faster clock to strobe the time intervals from transition to transition.

6.3.2.3 Modem, RF FSK (Method C). The third type of data transmission system is a true carrier system utilizing frequency shift keying. Bi-phase-L or NRZ-L data modulates the carrier, shifting it up or down in frequency. The modulated signal is AC coupled to the transmission line. The transmission line is terminated in its characteristic impedance for minimum RFI and driven with low power. Filtering is less complex.

Clocking techniques for this type of system are of the same possible types as those for the basic or Bi-phase system (A, V). The main advantage of this system is that the transmission spectrum is limited to a band centered about the carrier frequency. This allows placement of all data transmission in a low noise portion of the spectrum on the spacecraft thereby reducing the interference susceptibility to the lowest possible. The circuit complexity of this system is greater than that required for method A or B. A comparison between the three methods is illustrated in Table 6-2.

Table 6-2. Transmission Methods

Basic System

- Transmit Data Directly, Bit Sync Transmitted Separately
- DC Coupled to Line
- Transmission in Highest Noise Region
- Simple Filters
- Equipment Simplicity

Modem, Bi-Phase Level

- Transmit Combined Data and Clock
- AC Coupled
- Transmission in Medium Noise Region
- Fairly Complex Filters
- Medium Equipment Complexity

Modem, Carrier

- Transmit Data on a Different (Carrier) Frequency
- AC Coupled
- Transmissions in Low Noise Region
- Filtering Less Complex
- Most Hardware

6.3.3 Baseline Method of Data Transmission

Method B, baseband transmission of Bi-phase level encoded data (Manchester) was selected as the baseline approach for the I/O bus. By careful and proper design, this system utilizes a minimum of hardware for the maximum economy and reliability. At the fairly low data rates for the I/O bus, this method of data transmission presents little risk.

Method A has numerous disadvantages which outweigh its fairly simple implementation. DC coupling to the data link is probably the major disadvantage of this technique. It is, however, feasible to employ this type of data link for internal computer bus structures of short length.

Method C has one distinct advantage and is certainly a viable alternative to method B even with the added hardware it requires. The ability to move the data transmission band into a low noise region of the frequency spectrum may be the only reasonable solution if the spacecraft is "noisy". To specify this method now would proably be over-design, but by the same token, it can't be completely ruled out. Receiver filters for this carrier system are also much easier to design and require smaller components than for method B. The required signal power for this method should be lower than for method B due to the smaller margin allowed for noise.

6.3.4 Data Link Transmission Cable

Data links within the spacecraft require relatively short transmission paths. The electrical requirements of low signal attenuation, low delay distortion, and wide bandwidth are easily fulfilled for data links by existing cables. Choice of cable based upon these transmission requirements alone is very broad. However, other requirements can greatly restrict the cable choice. The cable may be subjected to and should operate satisfactorily in hostile noise environments. Also many mechanical constraints prevail, such as weight, strength, flexture, operation in the physical environment, size, configuration, physical limitations on connectors, etc. In addition, cost may be a factor.

The data cable cannot always be ideally routed within the station. It may share ducts and be cabled with wires of other electrical services. Routing near high powered electrical apparatus or near electrically sensitive sensors may become unavoidable. Therefore, the susceptibility to and emission of interference of the cable becomes a serious problem.

From available data, no consideration was given to a single wire-ground return transmission system, since the conductive noise alone shall render this a poor transmission path at its best. Multiplexing data on power cables or on wires for other services falls in the same unusable category.

The use of shielded, twisted pair is considered an economical solution. However, experience in the field of communications has shown that the nonuniformity in the ordinary twisted pair, even when well shielded, has limited use in the transmission of complex high frequency waveforms. These types of cables are usually employed in high level or lower quality signaling where interference is no problem. High quality twisted pair shielded cables are a must if they are to be used for the I/O bus system.

The transmission of basic NRZ-L data imposes the most severe cable requirements. If the data continuously marks or spaces, transmission at DC is required; the transmission band is in the high conductive noise region. High data levels must be used to overcome noise. Low pass filtering may be employed for band limiting, but the high frequency cut-off must not unduly limit the data rise time.

Encoding NRZ-L data to Manchester improves the cable requirements. Manchester code does not require coupling to DC. The transmission band is in a less severe noise region. The band can be limited by filters, since only the encoded transitions need be preserved to reconstruct the clock and code. Reducing the (noise) band allows the signal level to be reduced with a consequential reduction in RFI.

Modulating an RF carrier with Bi-phase imposes the least restrictions on the cable. The transmission band is in a low noise region. No DC component need be transmitted. Band limiting can be employed.

Possibly, the best cables for the data link considering noise rejection and low emitted RFI are those specially designed for this type of service. A common one is the video pair presently produced for the commercial television industry. This cable is a special design, well balanced, doubled shielded, twisted-pair which has proved superior to good quality single wire coaxial cable as to conductive interference rejection. The main objections to this cable are its weight and the commonly available connectors which are relatively large and unwieldy. Two conductor coaxial cable or twinax is another good choice of cable. Connectors for this type of cable are more readily available and are of small size.

The use of a special light weight (approximately one pound per 1,000 feet) flat cable has been considered. Autonetics has developed and tested a cable of this configuration. The cable features an internal shield design, with an extremely small loop area. Laboratory tests have shown much merit in its ability to operate in a hostile noise environment. Connectors are not readily available for this type of cable at the present time. Table 6-3 is a compilation of data on some possible cable types. The preferred cable for this application is a high quality, medium impedance, twisted pair shielded cable similar to FVP2-19. The special light weight flat cables should also be considered especially if new developments and improved connectors become available.

6.3.5 Clocking Techniques

Various clocking techniques were outlined in the data transmission link section. The simplest technique was not included in that discussion. It would be to have separate clock lines from data lines, with the clock generated at the computer and continuously illuminating the lines, or with the clock generated at each transmitter. The latter technique has the simplest decoding technique while the former technique is almost as simple except for some compensation necessary at the computer to handle the skew and delays during SIU transmission.

In the first case, all clocking per bus would be dependent on the single clock source and driver at the computer end of the bus. The second technique would allow SIU responses after loss of the clock driver at the computer end, but this is only possible if commanded by the computer through a separate bus to the SIU.

Table 6-3. Possible Data Link Cables

	Type No.	Prop. Velocity	Z_o Ω	Cap pf/ft	Attenuation db/100' 1MHz	10	40	100	Jacket Mat/OD	Shield	Dielectric	Conductor	Temp Range	#/100"	Manuf/ Notes
TW	RG108A/U	66%	78	22.5	.9	2.5	5.5	8.0	NVC. 245	TC . 200	PE . 079	TC/2 #20 (7/28)	-55/+80° C	2.9	III
TR	RG141A/U		50	28.5	.3	1.1	2.2	3.7	FGB. 195	SC . 195	PTFE. 116	SUSH (S) #20 . 039		3.1	III
TR	RG142A/U		50	28.5	.3	1.1	2.2	3.7	FGB . 206	SC(2). 173	PTFE. 116	SUSH (S) #20 . 039		4.1	III
PL	RG371 (3-50)/U	65%	50	31	.35	1.2	2.5	4.2	Poly. 145	OHFC (Solid)	PE	#21		1.01	UC 1/8 BR
TW	BL-782	66.5%	78±5	19.6	.38	1.17			V . 245	TC . 185	PE. 081	TC/2 #20 . 039			TWC 1-1/4 BR
TR	TR-503	66%	50	29.8		1.3	2.6	4.3	PE. 285 PE. 179	TC . 214 TC. 146	PE. 116	C(S). 085	-55/+85° C		TWC 1 BR
TR	RG59/U	78%	75	17.3		1	2.0	3.2	PE. 315 PE	C C	PE	C(S)#20 . 039			B
TW	FVP2-19	66%	110			0.79			V . 402	SC(2)	PE	#19	-55/+80° C		CS, WE 2 BR

The three methods of clocking discussed in 6.3.2.2 are: (1) utilizing a derived clock at the receiver in a phase-lock loop (2) transmitting a higher frequency clock with the data or (3) providing a higher frequency clock source at each SIU.

The first method is fairly complex and more suited to simplex operation and not the half-duplex message burst type operation to be used on the I/O data bus. It can therefore be eliminated on these grounds. The comments above for a clock technique utilizing a single clock source apply to this second method as well. Separate clock receivers and filters are also required in both cases to receive the clock at each SIU.

The third method requires separate clock sources at each SIU. In all likelihood these clock sources will be required for LP operation anyway and as such already exist at the SIU interface.

These techniques can be then listed for comparison as follows and are shown in Figure 6-4.

1. Separate Clock Lines

    a. Single Clock Source
    b. Multiple Clock Sources

2. Derived Clock

    a. Single Higher Frequency Clock Source
    b. Multiple Higher Frequency Clock Sources

In terms of line drivers and receivers, given that LP clocks exist at each SIU, 2b requires none. 1a and 2a have the same number of clock receivers and drivers. 1b has twice as many line drivers and receivers as 1a or 2a.

Techniques 1a and 1b require twice as many data lines as 2a and 2b, but 2a and 2b require dividers and logic to derive the clock from the data.

The choice then rests between 1a (half the hardware of 1b) and 2b (much less hardware than 2a). 1a has the simplest clock reconstruction circuitry at the cost of twice the data lines. 2b eliminates the need for the extra data link at the cost of more complex clock deviation circuitry. 2b also has an identical design at both the computer-bus interface and the bus-SIU interface while 1a does not. 1a may require adjustable delays at various receivers as well as special compensation circuitry at the computer-bus interface data receiver.

The preferred choice of technique is 2b. This eliminates the extra data links and discrete and linear circuits (clock receivers) at each SIU in favor of additional digital circuits more compatible with LSI techniques. All data receivers and Manchester decoders are now identical throughout the bus system. This selection also reduces considerably the interface pins necessary at each SIU.

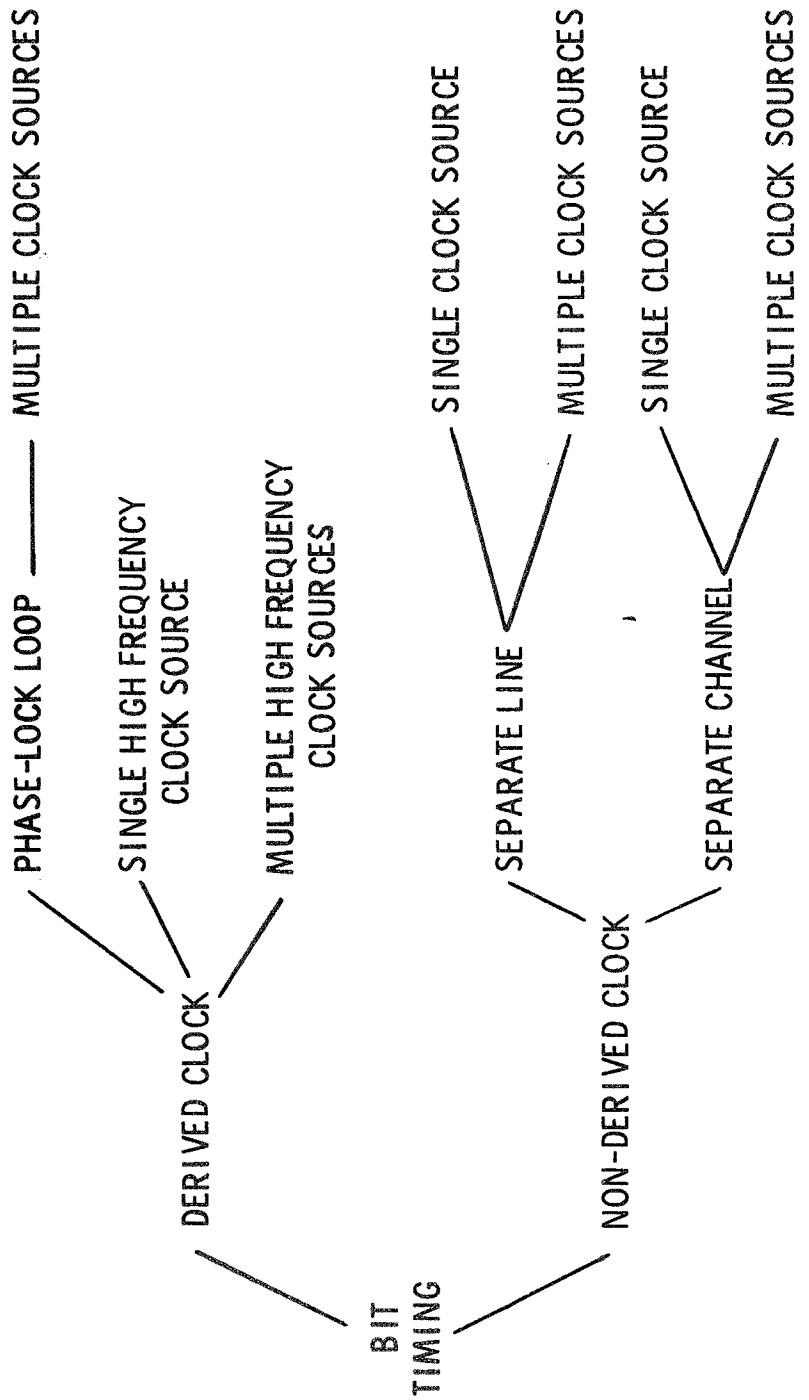


FIGURE 6-4. BIT TIMING TECHNIQUES

6.3.6 Synchronization Technique

Two types of synchronization are necessary for bus operation. The first type is bit synchronization. This establishes equal time intervals at both the transmit and receive ends of the data link. Bit synchronization will be achieved by deriving the timing from the received data stream in Bi-phase Manchester encoded form as discussed earlier.

The second type of synchronization required in the system is group synchronization. This is used to pinpoint an origin of time, which is the reference necessary to assemble detected bits into words and messages. There are two broad categories of synchronization techniques for group synchronization. One is to send a sync code as a part of the data stream and the second is to transmit a sync code in a different form from that of the data.

A request-acknowledge system has controlled message blocks and predetermined format. Few if any major synchronization problems result in this type of operation. No question of synchronization exists in the communication from LP to computer. The computer has requested this response and knows when it is to occur within less than one bit time. The only timing differences to be resolved are due to various line delays and skew problems. The sync code for this portion of the system can be extremely simple and minimal. A 3-bit Barker code[15] is proposed for this purpose, requiring a minimum of overhead. It will be transmitted by the LP in an identical fashion as data. The sync code will be /011/.

The synchronization at the LP to be achieved when the computer is transmitting occurs at somewhat random intervals. Before each of these sync codes occur, no data transmissions occur. By requiring this "dead band" interval all SIU's can be alerted that the next code broadcast will be a sync code. In this system operation the "dead band" interval can be quite large if the computer is not initiating any requests.

To shorten this interval and limit the time over which the SIU is "looking" for sync a "transmit no-data band" will be used.

This transmit band will be generated by the computer bus control. When it turns on the transmitter at the computer to initiate a message, the transmitter will first transmit a minimum of two zeros. This will then be followed by the sync code as data. The proposed code for this direction of transmission will be a second configuration of a 3-bit Barker code.

Thus at the SIU, a no transmit interval, followed by one or two zeros, indicates the time to look for the sync code. Two zeros should be enough to bring up the receiver (if not, more can be allowed) and ready it to sync detect. The sync code /100/ must then be detected exactly and correctly or the SIU will reset to a "wait for no transmit" interval.

The use of different sync codes for computer-to-LP and LP-to-computer transmissions rules out LP-to-LP communication. SIU sync detectors will not be able to detect the sync code generated by the SIU. This prevents SIU's from becoming active when another SIU is transmitting and going through an address recognition cycle, thereby possibly receiving false data. Additional safeguards are also present since the sync code is always followed by the LP address concerned with the communication.

An incorrect receipt of the sync code by an LP (at the proper time) also shall not elicit a response from the LP. This is an attempt to keep multiple LP's from transmitting simultaneously. Also, without correct reception of the sync code, the LP does not know when or whether a response is to be made.

6.4 ERROR PROTECTION TECHNIQUES

There are many techniques available for controlling errors in digital communications. In general, the necessary error protection for reliable communication is an engineering decision. The available techniques reduce the number of undetected errors to some reasonable number consistent with the design constraints of cost, size, weight, power, band width, etc. Beyond this point the expenditures necessary for error control produce diminishing results.

Of importance in any specification of error control techniques must be some realistic assessment of the expected cause of the errors to be controlled. Errors fall into many classifications (see ref. 6-12) and can be due to hardware, software, and/or external influences. The most difficult category to approach is errors due to external interference or noise. The resultant data after this type of error is of prime importance in selection of a means of error protection coding. For example, a simple parity check consisting of one bit can be used. This code is effective only for noise sources that produce an odd number of errors in the data word including the parity bit. If the expected noise characteristics are of this type, the parity code is very efficient. If, in fact, the errors do not follow this characteristic, the parity check is of little value.

Thus the first task of error control is to attempt to define the expected errors that can occur and should be at least detected. Also to be defined is the result on the system of undetected errors. This gives the design goal of the system in relation to the portion of errors that need to be controlled for the desired operation of the system.

This leads directly to the second task. Given that an error occurs, and it is to be detected to some confidence level, what action must be taken to minimize the effect of the error on the system. Error protection can be roughly divided into two areas, (1) error detection and (2) error correction after detection.

Error detection is by far the major technique employed in error protection. It is, of course, necessary to detect errors before error correction can proceed. But some systems can operate with little or no error correction, as long as error detection is used to eliminate erroneous operation utilizing bad data. Simple techniques such as retransmission of data, or no operation until the next valid piece of data is received, can be used for correction. The most complicated error protection schemes reconstruct the correct data (error correction) from the erroneous data (error detected).

Thus the impact of incorrect data at a systems input must be evaluated to determine the extent of error protection required for desired operation. Also the time allowable for corrective action must be defined. Systems which can tolerate loss of data for a period of time, or an interruption of service until appropriate action is taken, require a lesser degree of error protection techniques. Systems which cannot allow this type of event may require very complicated schemes to insure continuous correct data.

The following then need to be defined:

1) Sources, frequency, and types of errors expected in the system;

2) Number of errors against which system should be protected;

3) Confidence level for error protection;

4) Error rate allowable;

5) User system tolerance to errors;

6) Action desired or required after error detection;

7) Response time for corrective action;

8) Evaluation criteria to trade off various techniques that can achieve the desired results.

The following error protection techniques will be discussed in subsequent sections. The characteristics of each technique, along with the hardware and/or software implications will be examined. The advantages and disadvantages of the techniques will be related to the system constraints of size, weight, power, reliability, maintainability, and cost, where possible.

1) Parity Checking Techniques;

2) Complex Parity Checking Techniques;

3) Hamming Codes;

4) BCH Codes;

5) Fire Codes and Other Burst Codes;

6) Block Versus Convolution Coding Techniques;

7) Combinations of Codes;

8) Retransmission Techniques.

6.4.1 Parity Checks

A simple method of checking for errors is to use one redundant bit for this sole purpose. The value of this bit is so chosen that the number of "1's" in the group of bits to be checked is odd or even. These are consequently called odd parity or even parity checks. In this way, a single error in any one of the bits can always be detected.

Several important properties of parity checks are the following:

1) An error in the parity bit itself will be detected;

2) Parity check can detect errors to odd number of bits but may not detect errors to an even number of bits;

3) Parity check is invariant to shifting of the data if no bits are added or deleted;

4) Parity check is independent of the binary point location.

There are no overwhelming arguments in favor of either odd or even parity. Odd parity is usually used for two reasons. One: a string of bits including parity can never contain all zeros; and two: if the string had an even number of bits then the all 1's case also can never occur. These two cases can be used to detect circuit malfunctions that produce no output, or all "0's" output and all "1's" output. Neither parity check, odd or even, detects double errors by itself.

Parity checks need not include all bits of a group but can be selectively applied. Also, multiple parity checks may be used within a given group of bits or word, such that each parity bit checks some predetermined set of bits. Parity bits are the basis of most error detection techniques.

Single bit parity on a group of bits (word) is the most common and extensively used error control technique. It is extremely effective for random and independent noise sources when applied at a word level. It is the simplest technique in terms of hardware and can be performed in parallel or serial fashion.

6.4.1.1 Complex Parity Techniques. Multiple parity bits can be used to some advantage in coding for error control. One such parity technique is called modulo-4 parity to distinguish it from simple or modulo-2 parity techniques. Modulo-4 parity can detect some errors that perturb an even number of bits, but not all. It requires twice the number of parity bits per word or other group of bits being checked.

Modulo-4 has some advantages when detecting dependent errors such as for hardware faults, rather than random errors. For example, double errors or faults that tend to cause all erroneous bits to go to the "one" state or all bits to go to the "zero" state are detected.

6.4.1.2 Hamming Codes. Hamming codes are the simplest and most well-known of the error detecting and correcting codes. Since the publication of Hamming's paper describing this class of codes in 1950, more generalized codes have been constructed, of which Hamming codes are a special class. The application of Hamming codes to detect and correct errors of a random, independent nature are well documented.

The Hamming codes are quite easy to implement. The code is constructed by incorporating a number of parity checks for each word. Each parity bit checks the parity of certain groups of information bits within the word. The number of checking bits required for a certain word size can become quite large and this imposes a practical limit on the use of this code. For computer word sizes of 16, 24, and 32 bits, single error correcting or double error detecting Hamming codes require 5, 5, and 6 check bits.

Using Hamming codes adds more circuit complexity and costs more than simple parity checking. Hamming codes detect a larger number of errors for this increased cost and reduced coding efficiency.* They are certainly the simplest error correcting functions if so desired. Hamming encoding and decoding is usually performed in parallel, although serial operation is possible with modification or storage.

*Code efficiency is defined as the ratio of the actual information bits to the total number of bits used to effect transfer of these information bits, as a percentage of 100.

One disadvantage of Hamming codes is the inability to distinguish between multiple and single errors. For example, using the single error correcting or double error detecting code, if single error correction is assumed and a double error occurs, the correction will take place and form a valid code word but it will be an erroneous result. This action cannot be separated from, and will be identical to, that taken if only one error occurred. Thus one might be more inclined to use Hamming codes for error detection than corrections, especially where erroneous outputs are to be avoided.

6.4.1.3 BCH Codes. The binary BCH (Bose-Chauduri-Hoquenghem) codes are a generalization of Hamming codes for multiple error correction. They are as a class the best of the known constructive codes for channels perturbed by random, independent errors.[6-11] The error correction procedures of BCH codes are fairly complicated. Kastenholz[6-9] has investigated techniques for implementing these codes utilizing a digital computer and programming the decoding procedure. Large amounts of storage are necessary for practical code lengths as well as computer time.

The procedures involved in BCH error correction generally consist of solving for the roots of a t-degree polynomial and a set of t simultaneous equations (t equals the number of correctable errors). Recent work indicates that there may be ways of reducing this complexity.[6-13] The Berlekamp algorithm is one such technique, using either hardware or software, that speeds up the process of solving for the roots of the error polynomial.[6-4]

The use of BCH codes for error-detection only should also be considered. This part of the decoding process is by far the easiest and least time-consuming. The BCH codes are cyclic codes. Such codes are undesirable when loss of syncronism occurs since shifted cyclic code words are also valid code words. Other methods of detection are then necessary.

6.4.1.4 Fire Codes and Other Burst Error Codes. Conventional multiple error correcting codes will increase the reliability of data transmission at the cost of a relatively large increase in redundancy. Such codes do not make an efficient use of the fact that multiple errors are likely to be adjacent ones. Codes have thus been developed to handle this case and are called burst or non-independent error correcting codes.[6-1] Fire codes are a generalization of much of this work on codes to protect against the incidence of non-independent errors.

The Fire Codes compose one of the most efficient classes of burst error control codes.[6-9] Although Fire Codes are quite good they take a long time to decode and have a low efficiency for shorter block lengths.[6-4] The decode process can be simplified and shortened by eliminating some of the error correction capability of the code.

Fire Codes are oriented toward single burst errors per message over a fixed message size. Multiple burst errors are a good channel model but not many codes are known capable of correcting multiple bursts. Stone developed some efficient codes but they are difficult to implement.

Reed-Solomon codes are the most practical codes for multiple burst error correction. They are character based codes and a special case of BCH codes. The decoding for Reed-Solomon codes is complex and usually requires program implementation.[6-13] In general, they are simpler but similar in decoding complexity to BCH codes for the same block length.[6-4] The code efficiency is fairly attractive.[6-13]

6.4.1.4 Block Versus Convolutional Coding Techniques. In the preceding discussions of codes only block type codes have been considered. Convolutional coding techniques are not as well investigated or studied. The redundancy is usually higher in convolutional codes. It is possible to have a simpler decoding algorithm using convolutional codes and it is also argued that these codes are perhaps more adaptive to change in channel statistics. [6-4]

The methods of implementing these two basic types of codes differ substantially. These differences can be important enough to give either type of code an advantage in a particular application. Both types are subject to similar limitations and have approximately the same inherent capabilities. As far as theoretical error protection capability, there appears to be no significant difference between block and convolutional codes. [6-13]

Block codes have a simpler structure and are more useful for reasonably clean channels where low redundancy codes and simpler decoding algorithms can be used to satisfy the transmission requirements. [6-4] Since data is usually transmitted in blocks, block codes are better suited where error detection is required. [6-10] Convolutional codes can require considerable storage depending on block sizes and code lengths.

6.4.1.5 Combinations of Codes. It is well known that combination of codes, utilizing the advantages of each type of code, can be more effective than implementation of any of the individual codes. [6-11] Code combinations can take many forms, such as a different code structure for sub blocks or words than for the overall block. Many types of the previously discussed codes can be used together.

An example of this type of error control might utilize a random error protection code for part of the system and a burst error code for a different part. Each code would be matched to the expected channel characteristics for that portion of the system to be protected.

Various types of coding structures have been defined in the literature. Interleaved codes [6-13] are used to break up error bursts with subcodes interpreting the errors as independent. In this manner subcodes of lesser complexity can be used to protect against situations that would require codes with impractical decoding complexity. The disadvantage of these types of codes is the higher than necessary redundancy, and hence reduced efficiency, compared with the more sophisticated single code approach.

Two-dimensional [6-3] and N-dimensional codes are another class of error protection techniques that utilize the previously discussed codes or combinations of these codes. These structures are also termed interated codes. [6-11] The two-dimensional structure organizes data into rectangular blocks with separate (similar or dissimilar) codes applied to the rows and the columns. The decoding process can function independently, checking first the rows and then the columns, or simultaneous decoding can be done. The serial operation is much simpler and more practical than the parallel one. [6-3]

This type of coding allows identical or different codes to be applied to the rows and columns depending on the error statistics of the data or channel. Any two-dimensional array can be, of course, changed to a one-dimensional vector. In this form, a code with less redundancy (higher efficiency) could be obtained for the entire block. [6-8] However, as noted above, the combined decoding process might be simpler than a single more complex structure. Also this does not allow the flexibility for various codes and block lengths inherent in the iterated code technique.

6.4.1.6 Retransmission Techniques. Error detection is an attractive means of error protection provided it is possible to retransmit the data. In the case of data transmission systems this implies the existence of a bi-directional channel or some other feedback method to request the retransmission. Data links within a computer system can usually regenerate messages at the sending end when a message is discovered in error at the receiver.

If a feedback channel is present, one could calculate the probability of requests for retransmission and the average time the system operates in that mode given the noise statistics of the channel. Performance could be then evaluated in terms of efficiency or throughput. In general, detection and retransmission is effective against highly clustered errors.[6-13] For random error channels, or combinations of random and non-independent errors, some error will tend to appear regularly. In this event some minimum forward error correction can be used if necessary to improve the performance of the channel.

Benice and Frey[6-2] have examined the question of retransmission type systems based on an analysis that allows errors in the feedback channel and undetectable errors in both directions. Their investigation has shown the areas of relative superiority for both forward error correction and retransmission type systems based on throughput and undetected error rate. In almost all burst error channels and channels with a low probability of random errors, retransmission systems are shown to be superior. Only with an independent error channel with high error rates is the forward error correction method substantially better.

The redundancy of forward error correction is the additional bits added for this purpose. Using a detection-retransmission scheme the redundancy also includes all bits retransmitted. Therefore the efficiency of this technique depends on the detecting code redundancy as well as the probability of retransmission. With a low error probability, relatively long blocks or messages could be transmitted and fairly good efficiency achieved.

A somewhat increased error probability will increase the retransmissions. This can be overcome somewhat by decreasing the block size to decrease erroneous blocks. Further increases in error probability can shorten the block length so much that even the error detecting code is inefficient. At this point one might consider some simple forward error correction to lower the number of transmissions.

Detection-retransmission can achieve a large reduction in error rate with a modest amount of equipment. It is probably the most economical technique of error protection.[6-4] Also, it is pessimistic to assume that data is truly random, but no attempt is normally made to take advantage of any redundancy inherent in the data or in the data transmission process. In many cases discarding erroneous data has little or no effect, especially if a new data sample is to occur at frequent intervals. Thus one might only request retransmission after the loss of two adjacent (in time) data samples. Another possibility is to vary the handling of retransmission requests or size of retransmitted blocks depending on the priority or criticality of the user or the mode of operation of the system at that point in time.

6.4.2 Comparisons of Error Protection Techniques

The previous sections have detailed the various error control techniques, their applications, advantages and disadvantages. Each coding technique has, in general, been developed to satisfy a specific channel requirement or assumed error

problem. Within each code type exists a wide range of choices as to the size and degree of that type of code that can be applied to a specific situation.

The basic differences between most error protection codes relate to the type of errors to be detected and/or corrected. In order to compare these coding techniques for a specific application, the channel statistics must be known. These statistics are usually not available and thus models are used that approximate physical channels. The basic models for data channels are discussed in the next section.

6.4.2.1 Channel Models. An accurate determination of channel characteristics allows selection of the type and degree of error control coding necessary to insure the desired performance. This is the ideal situation. Practically, only estimates of channel statistics can be made, based on the kind of channel to be used and the possible external influences, without actually simulating the channel and its environment to great detail. Thus models have been developed which are gross approximations of possible channels. These models were conceived to allow analyses of the channel to be performed mathematically. Each model has a physical interpretation.

The basic channel model is called the BSC or Binary Symmetric Channel. The channel error statistics are independent of the binary symbol being transmitted, and each channel error is independent of all other channel errors. For a channel with 10 percent noise, the probability of a "1" being detected as a "0" is .1, as is the probability of any "0" being detected as a "1". This model satisfies the case where all errors are completely random and independent. Thermal noise in circuits can produce errors of this type.

Another channel model commonly used is called the burst channel. Disturbances within this channel occur within a span of bits called the burst length. Within the span the probability of error is high, and outside of the span the probability of error is zero.[6-4] This model is approximately true if one considers the probability of error outside the burst is very small compared to the probability of error within the burst.

This channel assumption is based on the fact that whenever an error occurred the channel is in a state where it is very susceptible to other errors within a short span of time. When the channel is not in an error state it continues error free until another burst occurs. In real channels this type of behaviour can occur with impulsive noise or switching transients.

Neither the BSC or the burst channel model are very satisfactory. Experimental results show neither model accurate for most practical purposes.[6-4] A third model, called a compound channel model,[6-4] was developed in order to present a more practical approach to actual channels. This channel model has two states. In one state it operates much like the BSC channel and in the second state like the burst channel. Thus multiple sources of error, both independent and dependent, are taken into consideration.

Other models have also been developed along the same lines as the compound channel. One is the low density burst channel which allows for bursts within bursts. A second type is the multiple burst channel which has proven to be fairly practical.[6-4] Reed-Solomon codes were developed for this model. The multiple burst channel errors occur in multiple bursts with each burst having a certain maximum duration.

6.4.2.2 Spacecraft Channel Model. A model to represent the computer communication channel for the space station will need to approximate the environment for good "wire" communications. The signal to noise ratios for hard wired channels tend to be high and can be adjusted as part of the design. Random noise is sufficiently low level that few errors are introduced from this source.[6-8] Impulsive noise is usually a more serious problem. Switching transients do exist on board but are usually unknown at design time. Certainly the basic BSC model is not adequate for this type of channel. The burst channel would probably be reasonable if only one specific type (or class) of impulsive noise was expected and nothing else. Neither model accounts for noise and errors due to faulty equipment and components.

The spacecraft system is also to be designed insensitive to all types of noise, not just specific noise sources. Certainly a compound channel model would be the minimum to be expected in terms of channel error statistics. The multiple burst channel is probably more practical yet and a good channel model choice. Unfortunately, good codes amenable to this type of channel are little known.

The compound channel model is probably the best model for the spacecraft bus system. It is flexible enough to include both random and burst errors from multiple sources. The additive effects of the error sources are probably "worst case" in terms of design for error control. This fits the highly reliable communication desired for the spacecraft data channel.

6.4.2.3 Error Protection for the Spacecraft Channel. A comparison chart of error protection techniques is given in Table 6-4. Given the compound channel model, the various coding techniques which are aimed at the control of random, independent channel errors are of little value if used by themselves. These include the simple parity checks, Hamming codes, BCH codes and others. The burst codes of Fire and others are also somewhat lacking for this channel model.

Thus the conclusion is, given a realistic channel model none of the so-called error protecting codes are really designed for this situation. One possibility that immediately comes to mind is the use of combinations of coding techniques. This offers some distinct advantages in the error control selection process.

It is also well known that a considerably lower error rate can be achieved with a given code technique by using error detection than by using forward error correction.[6-10] Another advantage of this approach is that most error protective codes are easier to implement for detection only than the more complex decoding for error correction.

Detection-retransmission techniques were shown to be superior to forward error correction when independent error rates are low and burst errors are expected. This situation is quite similar to the expected channel characteristics for hard-wired communications channels. The space station data bus is bi-directional and therefore amenable to this type of operation. The undetected error rate is of prime importance in the space station application. It is this parameter that is minimized with the error detection-retransmission scheme. Throughput is also maximized, which is not of overriding importance for this application.

TABLE 6-4. ERROR PROTECTION TECHNIQUES

TECHNIQUES	APPLICABLE CHANNEL MODEL	CODING AND DECODING IMPLEMENTATION	CODE EFFICIENCY	HARDWARE OR SOFTWARE REQUIREMENTS
SIMPLE PARITY	BSC	SIMPLE	HIGH	LOW
HAMMING	BSC	FAIRLY SIMPLE	MEDIUM	MEDIUM
BCH	BSC	FAIRLY COMPLEX	MEDIUM	HIGH
FIRE	BURST	COMPLEX	LOW	HIGH
BURST	MULTIPLE BURST	COMPLEX	LOW	HIGH
CONVOLUTIONAL	ANY	COMPLEX	LOW TO HIGH	HIGH
COMBINATIONS	ANY	SIMPLE TO COMPLEX	LOW TO HIGH	LOW TO HIGH
DETECTION-RETRANSMIT	ANY	SIMPLE TO COMPLEX	ADAPTIVE	MEDIUM

6.4.3 Baseline Method of Error Control

Taking into consideration the operational requirements and design considerations for the I/O data bus as detailed in the preceding section, a baseline method of error control is chosen. It is assumed that a good design practices will keep the bit error rate low (independent errors) for the bus.

The operation of the I/O data bus is based on a request-acknowledge system. Each communication from computer to LP is under computer control and is always bi-directional for proper operation. An error detection and request for retransmission type system will be the baseline method of error protection.

The bus communication takes place in blocks or messages to and from each SIU. Each message contains a variable number of words, i.e., each message is of variable length. All words will be of identical length. Each word will have error protection coding. The baseline method of error protection for the words will be the simplest of the various techniques, simple odd parity checks. This type of word protection will detect all errors to odd numbers of bits (independent or dependent) in each word. Utilizing this same format, a more complex method of error control can be applied for greater error detection if deemed necessary. Anything more than single bit parity will, of course, reduce the efficiency or throughput of the system and require either more hardware or software techniques.

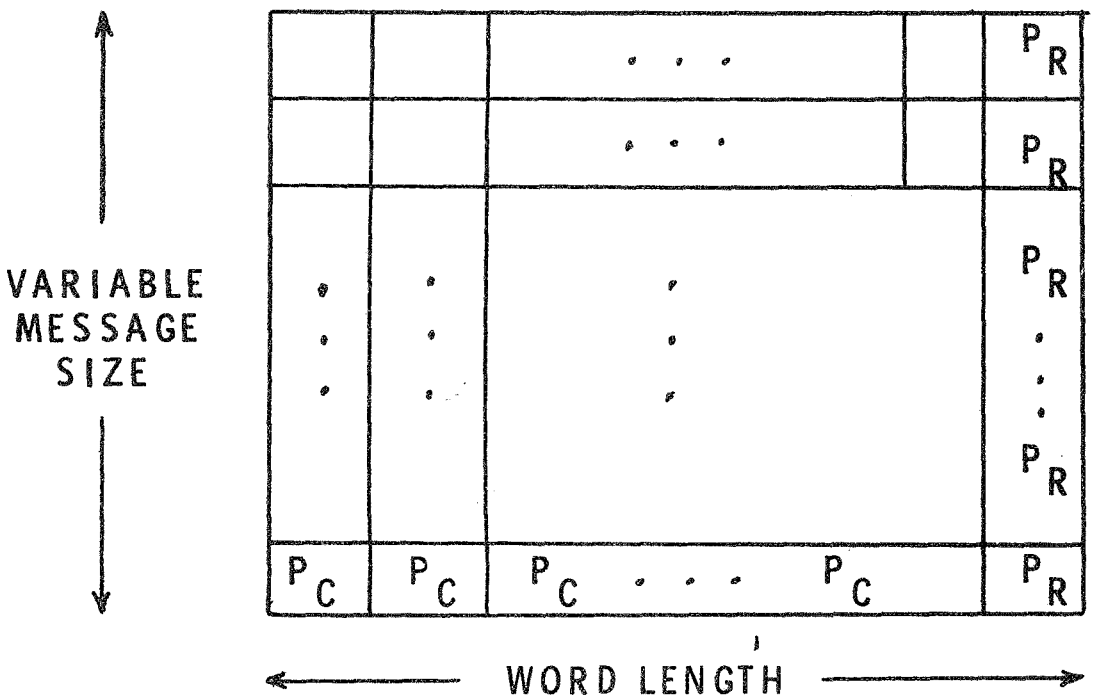
Each SIU also has provision for receiving multiple independent copies of each word and voting if necessary. This makes the baseline method even more attractive and allows some correction without apparent retransmission.

It is also proposed that the baseline method of error control includes a check on each variable length message. This can be extremely difficult depending on the coding technique utilized. The baseline method for this message check will be again simple odd parity. One parity bit will be sent that checks a single bit in all words of the message. Thus an additional word is transmitted at the end of each message containing only parity bits. This is a form of two-dimensional parity checking discussed earlier. It is illustrated in Figure 6-5.

Given the baseline method of error control, all errors in a message affecting three bits or less will be detected. Also all burst errors of word length or less will be detected by this method. (Bursts are defined as length b and not every bit within this segment must be in error.) To detect other possible errors, the two parity checks act in such a way that an undetected error can occur only when every erroneous word and every erroneous bit position contain an even number of error bits.[6-5] The total number of error bits must be an even number, four or greater, to cause a combined detection failure. The effectiveness of the message parity word increases with the size of the message; the fraction of all possible erroneous message blocks that will not be detected approaches $1/2^N$ for large N (N equals the number of words in message).[6-5]

The information word length has an even number of bits, thus with odd parity the all "0" word is an invalid word. The same check can be applied to the parity check word, i.e., that any message (excluding the checkword) containing an odd number of words cannot have an all "O's" in bit position i.

This baseline method of error control is quite simple to implement, and requires no predetermined knowledge of message length. The encoding and decoding can be done by software or hardware. The hardware implementation is quite simple and



P_R - ODD PARITY ON EACH WORD IN MESSAGE

P_C - ODD PARITY ON BIT i OF EACH WORD IN MESSAGE

FIGURE 6-5. TWO-DIMENSIONAL PARITY

very attractive especially in systems such as this where the data transfer is from parallel (computer memory) to serial (to bus) to parallel (LP memory) again.

This method of error control becomes less efficient when message lengths get shorter. For double word messages (control words) the efficiency is less than 66 percent. However, the error protection provided in this case is quite high. For critical items this gives maximum protection of the control word information.

The request for retransmission upon error detection is handled by the computer. The computer can thus determine the throughput of the system by the method in which it handles these requests, and can adjust for degraded modes of operation or subsystem criticality. It is also assumed that the computer can request the details of the parity check results to make its own determinations for fault detection and isolation. The two-dimensional simple parity check can also be used to correct a single error and simultaneously detect two errors to increase the system throughput. This might be done only at the computer (in software) where the cost is shared for all LP's involved.

Additional protection can be provided within this format for control information by judicious selection of the codes representing this information. Control field codes can be selected so that multiple bit errors are required to transform one valid control field code into another valid code, such as in the case of the reply bus code.

Most of the options available utilizing the baseline method of error control are dependent on actual implementation and hardware involved. Until an actual hardware design of the overall information transfer system is done these should rightly remain options. Given the design, there will probably be a number of errors due to specific hardware faults that one would want to specifically protect against. The flexibility of the baseline system allows for these design inherent areas to be error protected and/or isolated.

6.5 I/O BUS CONFIGURATIONS

The I/O Bus System in this report has been mechanized in such a manner as to allow operation in either of the two candidate system approaches. Voting, if necessary, has been assumed as part of the LP and not of the SIU. Each SIU has independent, redundant interconnections to up to four independent party lines.

Similarly, a voting configuration at each LP would in all likelihood change the basis for determining necessity of retransmission by the computer bus control (IOP). These changes are easily accommodated within the existing data transmission format specified for the bus system.

6.6 SUMMARY OF PREFERRED BASELINE MECHANIZATION

The I/O data bus will operate as four separate, independent communication links. The operation of each individual data bus will be under complete computer control in a request-acknowledge format. Communication on the bus will be in messages, each message headed by two control words and ending with a check word. The number of words in a data message is variable up to a maximum of 63 total words.

The baseline method of data transmission on each bus will be baseband, utilizing bi-phase level encoding (Manchester). The data links will be twisted pair shielded cables, balanced and terminated. Each SIU will be coupled via transformers to the data link and resistor isolated from the data line for short circuit protection. The data link will be a true party line configuration. Appendix 9 is a data transmission subsystem design for the GN&C system. Clocking for each SIU will be derived from the received data utilizing a higher frequency clock source provided for the LP. A data synchronization code will be used for group synchronization at each SIU. A different synchronization code will be used for communications to the computer ruling out any communication between LP's on the same party line.

Two-dimensional simple odd parity checks will be used on all messages for error protection. An error detection-request retransmission scheme will be used to ensure correct data transfer to each LP. The retransmissions will be under computer control and mode sensitive to allow full computer control of the bus system throughput. The possibility of simple forward single error correction plus double error detection is available with this error control format to improve throughput at any time.

Each SIU will have multiple I/O bus lines and circuitry to allow multiple independent receptions of the same data if desired. This level of hardware redundancy at the SIU also allows SIU reconfiguration after SIU or bus failures. The SIU design shall be such that no single SIU failure shall cause loss of any data link. Redundant, independent, transmitter enable circuitry and time-outs shall be used for this purpose. The SIU design and operation is included in the report on the Local Processor (Section 9).

7.0 MECHANIZATION OF SELECTED COMPUTER SYSTEMS

7.1 GENERAL

The evaluation of the candidate computer systems resulted in the selection of organization number 4, the restructurable multicomputer, with the voter-comparator-switch system concept, mechanized with conventional technology. This section of the report deals with details of the mechanization of the selected candidate.

The computer system previously shown in Figure 4-17 is a complex of four identical computers configured as a set of two multiprocessors and tied together by data links. Each computer will be a general purpose digital machine operating on an internally stored program. Table 7-1 lists the major features of the computer.

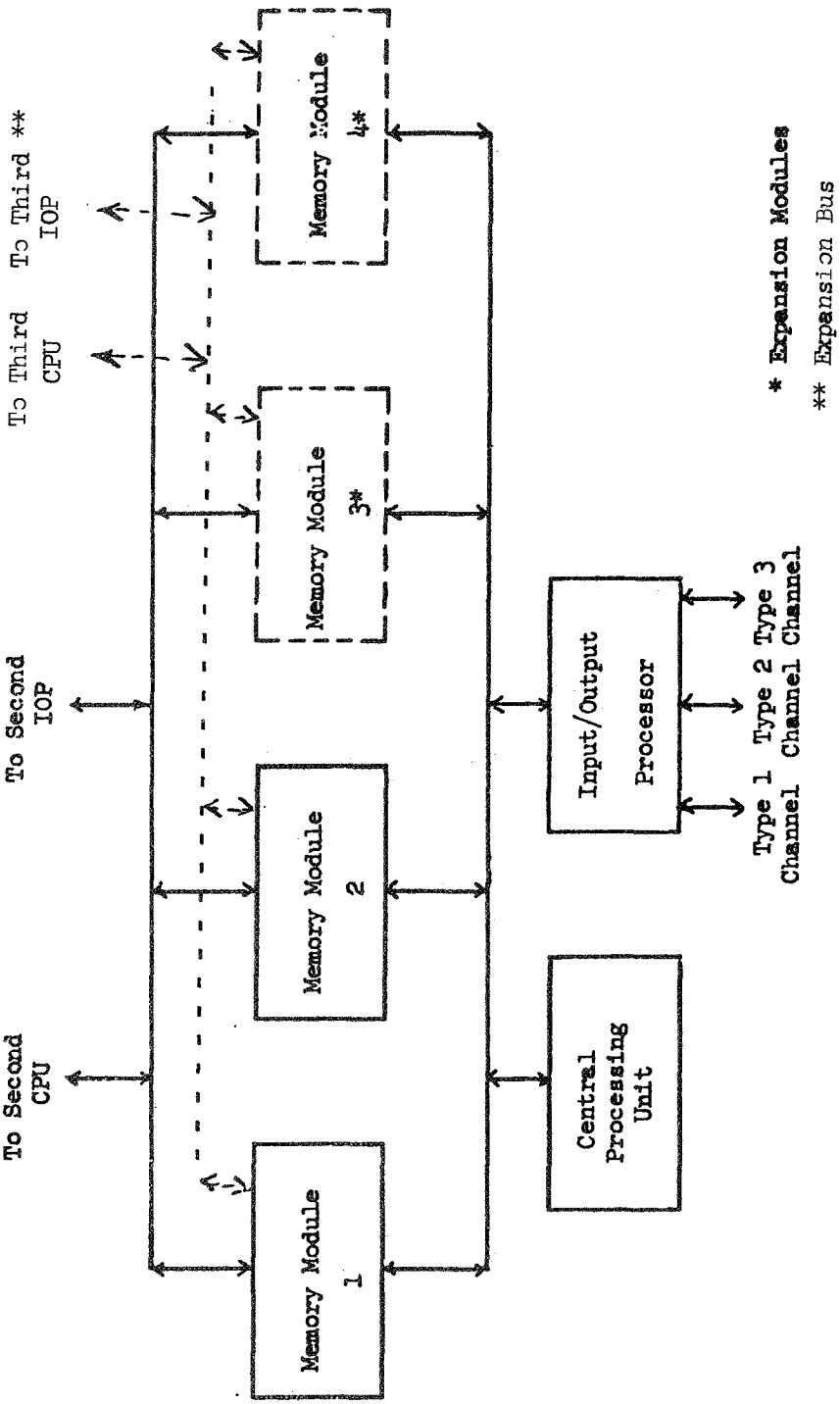
The computer will consist of three operating areas:

1. Central processing unit (CPU);
2. Memory;
3. Input/Output Processor (IOP).

The CPU will perform arithmetic and logical operations on data according to the stored program. The IOP will perform the task of handling communications with devices external to the computer. The VCS device will be imbedded in the discussion of the IOP. The memory will provide non-volatile storage for permanent and temporary data and for programs. Figure 7-1 is a block diagram of the computer. The block diagram of the total computer system was shown in Figure 4-17.

Table 7-1. Computer Major Features

Data Word Size	16 or 32 bits including sign
Instruction Word Size	16/32 bits
Logic	Dynamic MOS, 4 phase, 1 megahertz bit rate
Memory Type	Plated Wire
Memory Size	32,768 words (32 bits) (expandable to 65,536)
Input/Output	Operates independent from CPU on internally sorted program, Serial and Parallel channels, Serial Data Rate: 1 megabit per second; Parallel Data Rate: 250K words per second.
Special Features	Real Time Clock Floating Point Majority voting or comparison mode on output to local processor subsystems via VCS function



COMPUTER BLOCK DIAGRAM

FIGURE 7-1

7.2 FUNCTIONAL DESCRIPTION

7.2.1 Central Processing Unit

The CPU will execute a program stored in the memory. This program will consist of instructions of the following types:

1. Arithmetic (add, subtract, multiply, divide, etc.)
2. Logical (and, or, complement)
3. Shift
4. Data moving (load, store)
5. Branch (conditional, unconditional)

The CPU will have the capability of single precision, double precision, fixed and floating point arithmetic operations. The CPU will be capable of addressing the memory directly, indirectly or by indexing. All operations will be done in parallel except the shift operations which will be serial.

7.2.2 Memory

The memory will be made up of independent modules each consisting of a plated wire array and electronics for reading and writing in the array and for control functions. The basic memory configuration will be two modules and the maximum configuration will be four modules. The characteristics of each memory module will be:

1.	Word Length:	32 bits
2.	Capacity:	16,384 words
3.	Type:	NDRO
4.	Cycle Time:	1 microsecond
5.	Parity:	1 bit for each 16 bits

Each module will be capable of interfacing with three parallel buses. Each bus connects the module to an IOP and a CPU. Operations over the buses will be controlled by the memory module electronics. Service over the buses will be on a round robin priority basis unless configured in a special mode by the lockout electronics.

Data will be read from the memory in 32 bit words. The requestor will decide whether to use part or all of the word. Data will be read into memory as full words or half words. Either half of the word may be written into with the other half not being disturbed. The requestor will control the type of memory write wanted.

7.2.3 Input/Output Processor

The IOP will control the flow of data between the computer and the rest of the system. The IOP will have three types of channels:

a. Type 1 Used for computer to computer data flow;

b. Type 2 Used for data flow over the data I/O bus to equipment outside of the computer complex;

c. Type 3 Used for block transfer of data with the mass memory complex and data management system.

The type 1 input/output section will consist of three input channels and one output channel. Each of the input channels will be dedicated to a specific computer. The output channel will be connected to all other computers. The input and output channels will be able to operate independently and simultaneously. Data will be sent bit serial and word serial over these channels.

The type 2 input/output section will consist of one input channel and one output channel. The channel will have a dedicated cable to connect the computer with the local processors (LP) in the subsystems in the spacecraft. Data will be sent over this channel in a bit serial and word serial fashion.

The type 3 input/output section will consist of a single bi-directional channel consisting of 17 data lines and two control lines. Response to requests on this channel may be inhibited by the setting of a flag in one of the IOP commands. Data will be sent over this channel in a bit parallel and word serial fashion.

Transmissions over the type 1 output and type 2 channels will be initiated by the IOP executing commands from a program which is stored in the memory. Transmissions over the type 3 channel will be initiated solely in response to external requests. The IOP will be capable of operating the computer system in a voting, comparison or non-redundant mode by means of the VCS functions described previously in Section 4.5.

7.3 MECHANIZATION OF INTERNAL MODULES

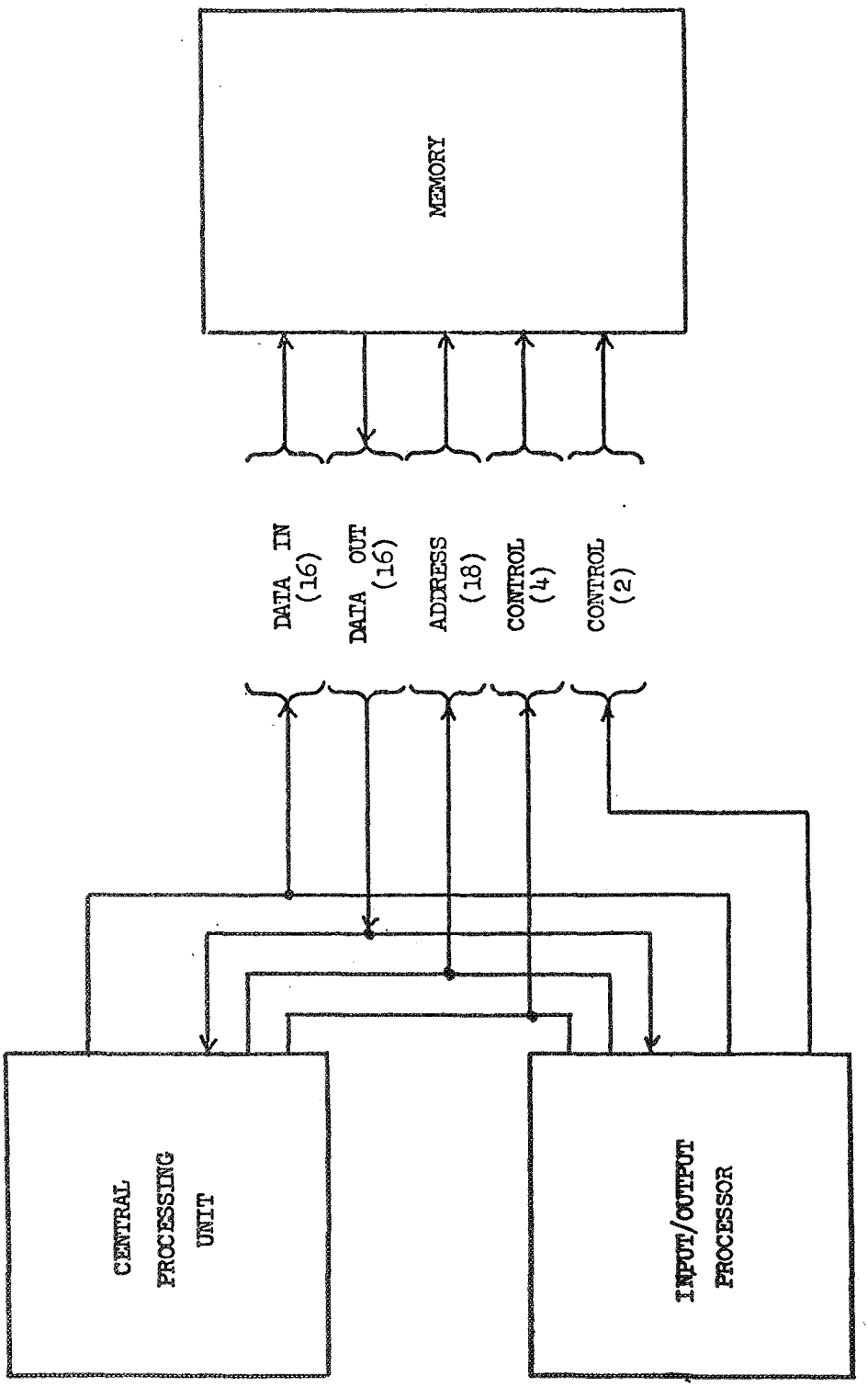
7.3.1 Computer Memory Bus

The computer memory bus provides the communication path between the three functional areas of the computer (CPU, IOP, and Memory). Traffic flows between the CPU and the memory or between the IOP and the memory. There is no direct path between the CPU and IOP. A block diagram of the bus is shown in Figure 7-2. A memory interfaces with three buses while the CPU and IOP each interface with only one bus.

The bus consists of the following:

1.	Address lines	18
2.	Control lines	6
3.	Data-in lines	16
4.	Data-out lines	16

The bus is operated at a ten (10) MHz rate in order to maintain the effective memory access time as seen by the CPU and IOP at a nominal one microsecond.



COMPUTER MEMORY BUS

FIGURE 7-2.

The address lines are used to carry the code specifying the memory module (4 bits) and the desired location (14 bits) in that module. The given numbers represent the largest computer configuration that is possible (16 - 16K modules in each compartment with this addressing scheme).

The control lines are used to control memory operation. Two of these lines carry signals representing the operational status of the two computers which are physically co-located in the compartment. Each line is driven by one diagonal term of the P matrix from the VCS of the IOP assigned to one of the computers. The other four control lines are used to carry the codes that represent the various memory commands.

Separate data-in and data-out lines are used to maintain maximum bus speed and simple circuit arrangement. Savings in circuit complexity are derived by using a half-word byte channel instead of a full-word channel. The CPU will utilize both half- and full-word formats while the IOP will use full-word format only. Full words are to be sent as two half-word bytes 100 nanoseconds apart.

The IOP and CPU share the use of the memory bus; access to the memory bus will be handled on a first-come-first-served basis. However, since the IOP and CPU programs areasynchronous with respect to each other, some provision must be made to eliminate conflicts due to simultaneous requests for the bus. The IOP sends and receives data in serial and continuous fashion. To prevent any disruptions in the messages and to reduce hardware buffering requirements, the IOP is to have priority over the CPU in the event of simultaneous bus requests. This priority assignment is performed by the IOP. The CPU must send a signal to the IOP requesting bus access. The IOP determines if the IOP needs the bus at that time or is presently using the bus and sets a bus access ready line true or false accordingly. As soon as the IOP releases the bus, the CPU will be allowed to use it.

7.3.2 Central Processing Unit

7.3.2.1 Introduction. This section describes the central processing unit (CPU)of the computer. The features described are the minimum required to perform the computational task as seen in this study. The details of some features (e.g., number of registers or interrupts, number and exact types of instructions) will be subject to change as a result of studies now in progress or studies that are to be made in the future.

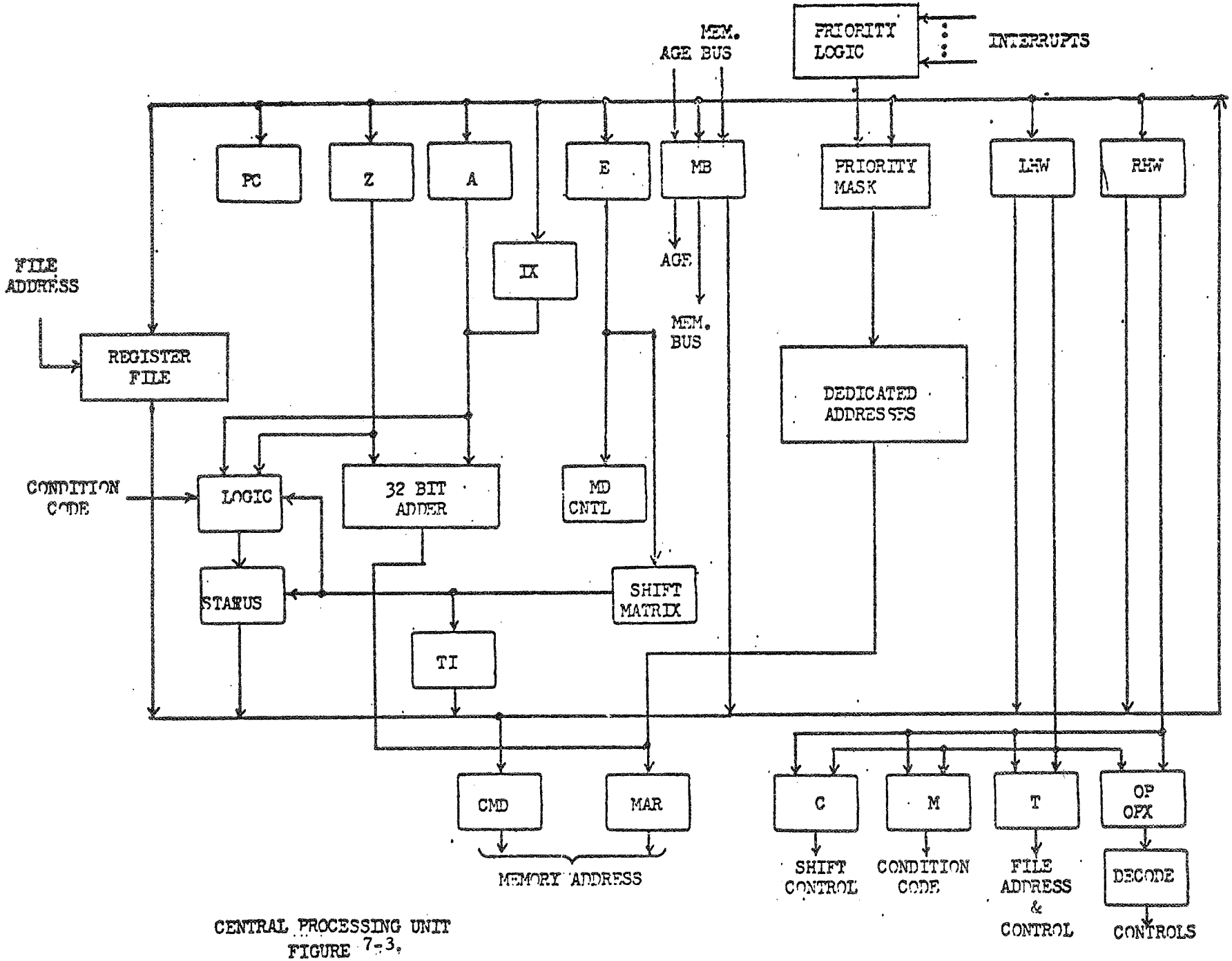
7.3.2.2 General. The CPU is a 32 bit, parallel operating unit using general registers in its instruction implementation. The instruction set provides for 16 bit half-words, 8 bit immediate operands, and 32 bit words.

Memory addressing is by full word, that is, every 34 bits (32 data + 2 parity) of memory has an address. The 18 bit address may access any one of 196,608 words of memory (12 modules, 16,384 words/module; 4 modules/CPU). A memory word is read out to the CPU 16 bits at a time and assembled into a 32 bit word. A buffer assembles full-word instructions from two half-words. Data may be 16 bits in length (half-word) or 32 bits (full-word).

Status control words (SCW) are used to define the status of the computer. These contain interrupt status information, comparison results and condition codes, and instruction address. SCW's are established in fixed memory addresses and used in the execution of interrupts.

7.3.2.3 CPU Organization. The CPU organization is shown in Figure 7-3. Functions of the registers and logic blocks are defined in the following:

Register/Logic Block	Usual Function
Register File	Provides storage for general registers
PC (18)	Program Counter
A (32)	General Purpose Data Register
E (32)	Extension Register
MB (32)	Memory Data Buffer
Priority Logic	Interrupt Sensing Logic
Priority Mask	Interrupt Mask Logic
LHW (16)	Instruction Left Half Word Register
RHW (16)	Instruction Right Half Word Register
MD CNTL	Multiply-Divide Control Logic
MAR (18)	Memory Address Register
OP (4), OPX (4)	Operation Code Register
Z (32)	Auxiliary Operating Register: This register is used to hold intermediate results during floating point and indexing operations.
IX (18)	Indexing Storage Register: This register is used to hold numbers used for indexing of instructions.
TI (32)	Temporary Buffer: This register is used for counting of shifting during normal shift and normalizing operations.
T (9)	File Address Register, Decode Logic and Count: This register is used during register to register operations to keep track of the register being used.
M (4)	Condition Code Decoder: This register is used to store results of comparison operations and to control the logic concerning operation as a result of the comparisons.
C (5)	Shift Control Logic: This logic is



CENTRAL PROCESSING UNIT
FIGURE 7-3.

Register/Logic Block	Usual Function
	used to control all shift and normalizing operations.
CMD (2)	Computer Memory Designator: This register is used to hold the two bit code that specifies which group of four memory modules is to be accessed over the memory bus.

Communication with the memory is accomplished through MAR and MB. The MAR is provided information regarding memory address from the adder, the AGE and from the dedicated address matrix selected by the enabled interrupt. Data and instructions are processed through the memory buffer register, MB.

The memory addressing for obtaining instructions and operands is generated through use of the PC, Z, A and IX registers. Further, the Z, A, and E registers are used for the arithmetic functions. Thus, the single 32 adder serves both functions. Also to be noted are the logic blocks to perform shifting and the operations of And, OR, Exclusive OR and Comparison. The latter logic, together with the Adder and Shift Matrix outputs, is used to generate the 32 bit status control word. The Register File is a high speed scratch pad memory used for index register storage as well as general register operations. Each register is 32 bits long.

7.3.2.4 Interrupts. The interruption system enables the change of the state of the CPU as the result of conditions occurring in the CPU or external to it. An interrupt is serviced when the preceding instruction is finished and the next instruction not yet started. The interrupt causes a dedicated address definition and information transfer to and from memory locations. The initial dedicated address is used to store the current contents of the Program Counter. The dedicated address is then incremented and a word obtained from the next full word location. This word contains the address of the start of the interrupt subroutine. The interrupt action requires a write and a read memory cycle.

Both external and internal interrupt capability is provided. A priority interrupt system provides program control for rapid response to special external and internal conditions.

Each priority interrupt is assigned a fixed address in memory which contains the linkage information to obtain the start of the corresponding subroutine. Upon completion of that routine, return to the original program is possible through the restoration of the original contents of the program counter. Further, each level of priority interrupt can interrupt lower levels. Should a higher priority interrupt occur, new locations are defined for status retention. The highest priority program will be processed and return to lower level operation accomplished automatically.

An interrupt masking capability with loading and reading instructions is provided for inhibiting various interrupts and status monitoring. Masked interrupts remain pending until unmasked and taken. This capability permits reassignment of priorities in real time.

7.3.2.5 Instruction Formats. The CPU will utilize eight basic instruction formats. These eight basic instruction formats are denoted by the format codes RR, R, S, PC, RS, RSX, RRS, and SI. The format code expresses, in general terms, the operation to be performed.

1. RR, denotes Register-to-Register operation;
2. R denotes Register operation;
3. S, denotes a shift operation;
4. PC, denotes a Program Counter operation;
5. RS, denotes Register-to-Storage operation;
6. RSX, denotes Register-to-Storage-Indexed operation (indirect option also available);
7. RRS, denotes Register-Register-Storage operation;
8. SI, denotes storage and immediate operand operation.

Formats 1 through 5 are half-word format instructions and formats 6 through 8 are full-word format instructions.

For addressing purposes, operands can be grouped in three classes: explicitly addressed operands in main storage, immediate operands placed as part of the instruction stream, and operands located in the general purpose register file.

To permit the ready relocation of program segments and to provide for a flexible specification, all instructions referring to main memory have the capacity of employing a full address. This address used to refer to main memory is modified by the following:

Base Address (B) is a 16-bit number contained in a general register specified by the program in the B field of the instruction. The field is included in every address specification. The base register shall be utilized as a means of relocation of programs and data. The base addressing shall provide for addressing all of main memory.

Index (X) is a 16-bit number contained in a general register specified by the program in the X field of the instruction. It shall be included only in the address specified by the RSX instruction format.

In forming the address, the base address and index are treated as unsigned 16-bit positive binary integers. The two are added as binary numbers, ignoring overflow.

Examples of instruction types in the various formats are as follows:

1. RR Format
   - Add Registers
   - Subtract Registers
   - Multiply Registers
   - Divide Registers
   - Logical And
   - Logical OR
   - Logical Exclusive OR
   - Load Registers
2. R Format
   - Branch on Condition
   - Branch on Condition, Return Status
3. S Format
   - Shift Left
   - Shift Right
   - Cycle
4. PC Format
   - Branch on Condition
5. RS Format
   - Add HW
   - Add FW
   - Subtract HW
   - Subtract FW
   - Multiply
   - Divide
   - Store HW
   - Store FW
   - Load HW
   - Load FW

6. SI Format
   - Compare Immediate
   - AND Immediate
   - OR Immediate
   - Exclusive Or Immediate

7. RSX Format
   - Add HW
   - Add FW
   - Subtract HW
   - Subtract FW
   - Multiply HW
   - Multiply FW
   - Divide
   - Load FW
   - Store HW
   - Store FW

8. RRS Format
   - Load Multiple
   - Store Multiple
   - Branch on Register Condition

7.3.2.6 Read Only Memory. When the computer fails, as indicated by the P Matrix of the VCS in the IOP, a program is to be executed by the CPU to determine the functional area that has failed. This program cannot be stored in main memory because the failure may be in the memory module or the memory bus. A fault isolation program will be stored in a read only memory (ROM) in the CPU. This program is for internal reconfiguration only; external reconfiguration on the computer system level is accomplished by the non failed computers.

The ROM will be a semiconductor memory with a capacity of approximately 8192 bits. The program will be stored as 32 bit words. All addressing and readout circuitry required for the memory will be part of the ROM package.

7.3.2.7 Arithmetic Formats. The CPU operates in either floating or fixed point arithmetic in the two's complement number system. Data can be either half-word or full word in the fixed point operations. Data in floating point operations are specified in two parts (mantissa and characteristic) occupying a full 32 bit word. The mantissa (or fractional part of the number) is stored in 24 bits of the word and the characteristic (or power of two multiplier of the number) is stored in seven bits. The sign bit makes up the thirty-second bit of the word.

7.3.2.8 Memory Bus Interface. The CPU interfaces with the main memory by means of the memory bus. The bus was described in Section 7.3.1. The address sent over the address lines of the bus is developed in the CPU by automatically appending the two bits of the computer memory designator register to the most significant end of the sixteen bit address derived from the instruction or from the program counter when a memory module is accessed. The computer memory designator register is loaded by the CPU program permitting full addressing capability for the maximum number of memory modules (12) in one compartment (theoretically 16 can be addressed however each CPU or computer is designed to power a maximum of 4 modules and a maximum of three busses or computers may be provided in one compartment).

7.3.3 Memory Description

7.3.3.1 General Description. The computer will utilize up to four (4) random access 16K word, 34 bit plated wire memory modules for the main storage function. Read and write cycle times will be 750 nanoseconds and 1.0 microsecond, respectively. Read access time will be 500 nanoseconds for the first half-word and an additional 100 nanoseconds for the second half-word.

Each memory module is composed of a 16,384 word plated wire array organized internally as a 1024 x 16 multiword (2-1/2 D) system. This type of organization reduces the word access lines and circuitry by adding a third selection dimension. Word addressing in the 16,384 word stack is accomplished by a selection of one of 32 "X" positions and one of 32 "Y" positions, thus isolating one of the 1024 word lines of the chosen stack. A "Z" selection of one of the 16 words along the selected word line completes the addressing. This operation involves the low-level switching of the proper input for each of the 34 sense amplifiers during a read cycle or driving the proper lines with bit current for each of the 34 bits in the word during a write cycle.

This type of organization minimizes the required circuitry and yields an efficient and reliable system with correspondingly less stand-by power. The multiword organization (16 - 34 bit words on each word line) is possible because of the plated wire properties of equal drive word current for both read and write as well as the nondestructive readout.

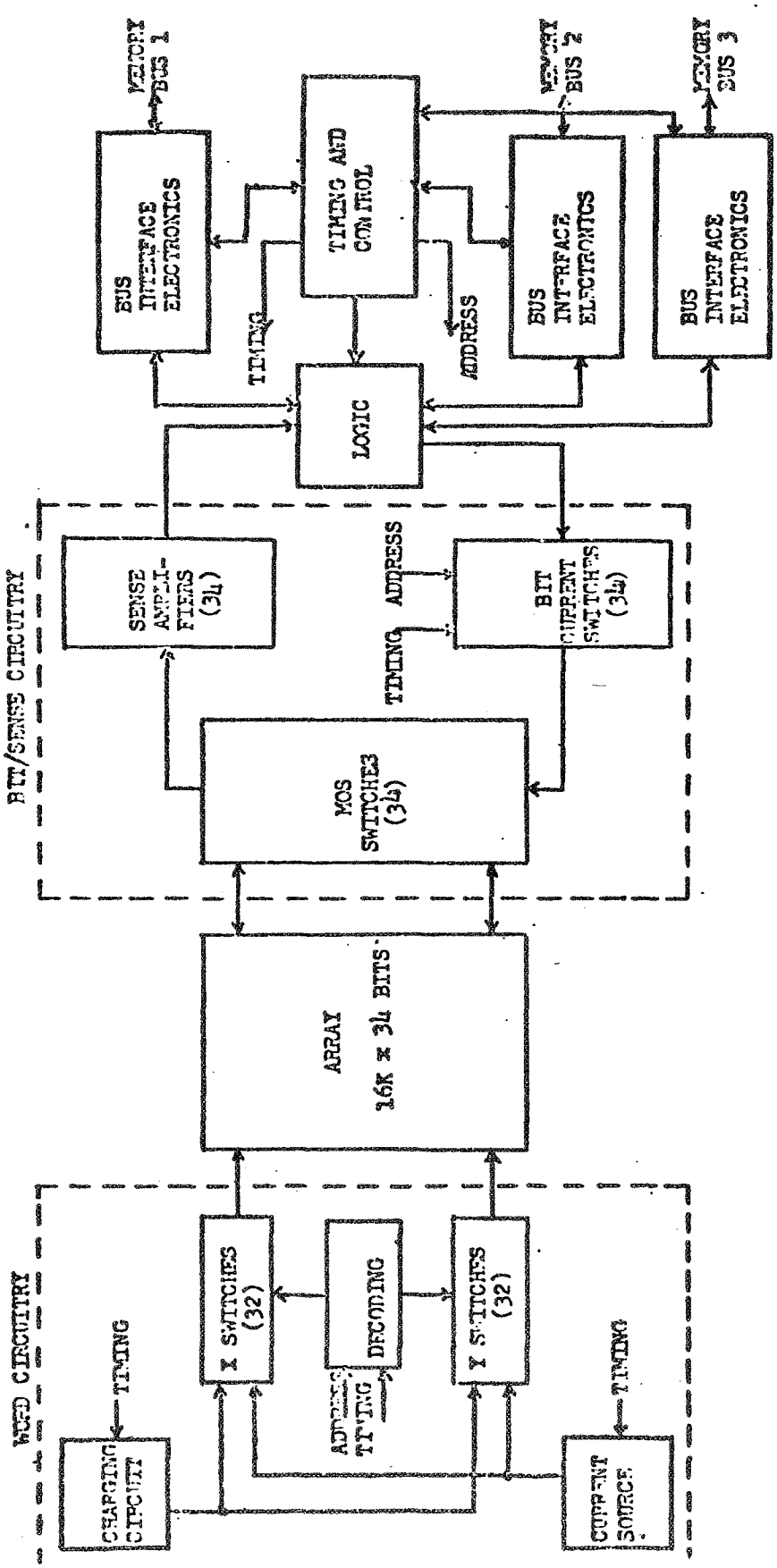
7.3.3.2 Functional Block Description. The memory may be broken down as shown in Figure 7-4.

7.3.3.2.1 Word Circuitry. The word access circuitry performs the following functions during both Read and Write cycles:

1. Makes the 1 of 32 "X" selection and the 1 of 32 "Y" selection resulting in the required 1 of 1024 word line selection for the chosen stack.

2. Provides the required Read and Write word drive currents.

3. Supplies array charging current to force recovery of the memory word line bias voltage between memory cycles.



MEMORY FUNCTIONAL BLOCK DIAGRAM

FIGURE 7-4.

7.3.3.2.2 Bit/Sense Circuitry. The bit/sense electronics provides bit drive to the proper one of 16 bit lines (as determined by the address) for each of the 34 bits of the requested word during a Write cycle. During a Read cycle, the appropriate one of 16 bit lines is selected as an input to the sense amplifier for each bit in the requested word. The sense amplifier then provides amplification and discrimination before outputting data at a suitable logic level.

The circuitry consists of decoder-low level switch devices, sense amplifiers and bit current drivers.

7.3.3.2.3 Memory Array. The array provides the NDRO storage elements required for the system. A 16,384 word block consists of two dual planes of plated wire mats laid out in a one crossover per bit arrangement plus the word isolation diodes and low level switch devices.

7.3.3.2.4 Timing and Control. The timing and control circuits provide the required internal timing, decoding, temporary storage and buffering required to make the memory module function. It consists of logic in conjunction with a tapped delay line being used as a central timing element.

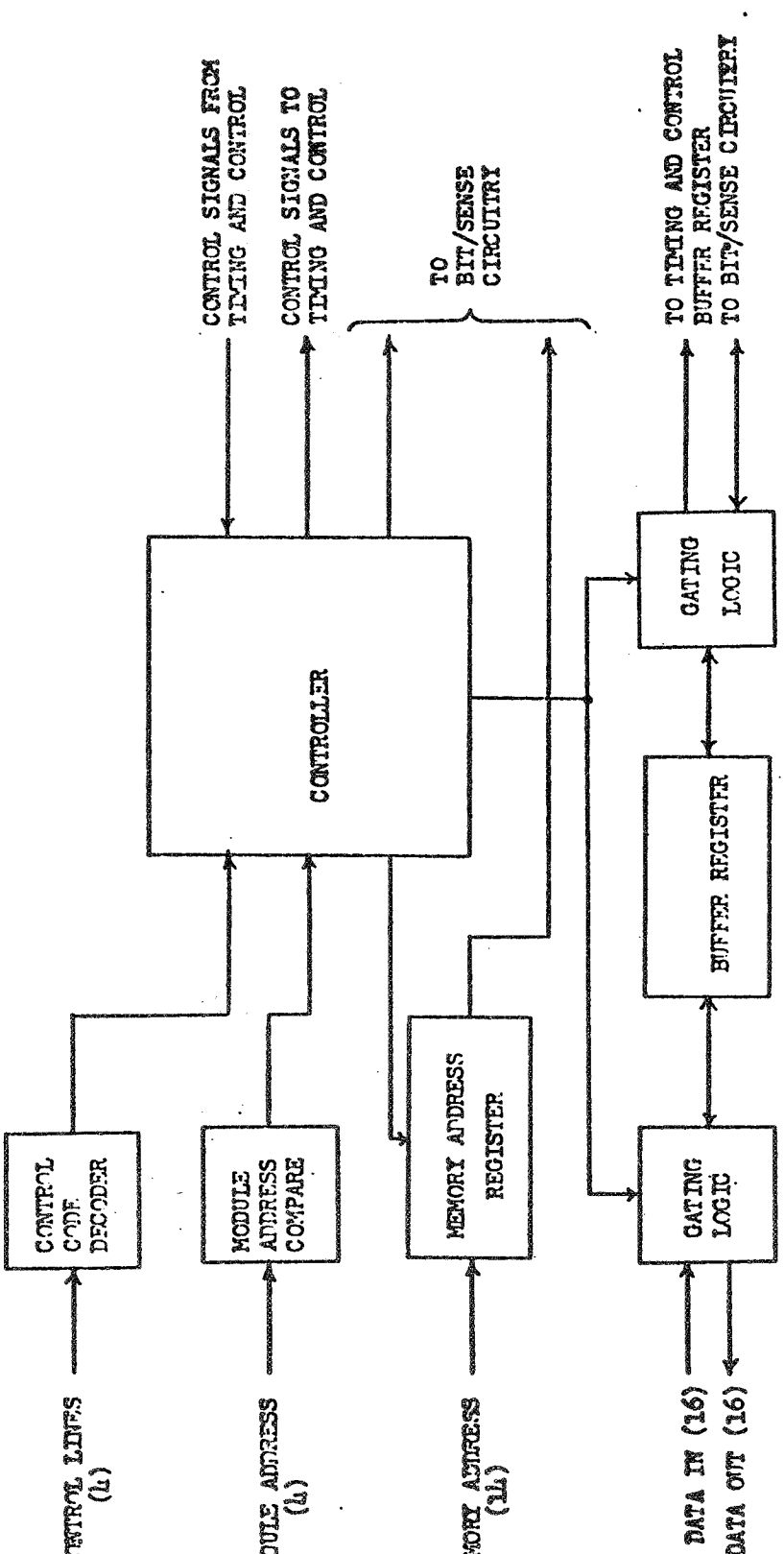
Operations are determined by decoding the control and command words received over the memory buses from the CPU or the IOP.

7.3.3.2.5 Bus Interface Electronics. The bus interface electronics provide the necessary buffering to match the memory buses with the internal circuits of the memory module. Also provided is the module address comparator to determine whether data on the bus are for the particular memory module.

7.3.3.3 Memory Module Control. The memory module mode control is located in the timing and control (TAC) section while the address and control code decoding is done in the bus interface electronics (BIE) section. Both sections control access operations to the array.

7.3.3.3.1 Bus Interface Electronics Section. The BIE section performs the functions of address detection, control code decoding and read/write operations for data sent over the buses. The memory module has three BIE sections, one for each bus. A block diagram of a single BIE section is shown in Figure 7-5.

7.3.3.3.1.1 Normal Operation. Each memory module of the computer is assigned a four bit address code. The module will accept only those control codes that accompany an address code whose four (4) most significant bits are the same as the module address code. The module address code is held in the module address register of the BIE section. The address code is also stored in a dedicated location in memory. Upon recovery from a power transient, the module address code is accessed from the array and placed in the module address register to return the module to the pre-transient condition. Each BIE section may have a different address, so all codes must be stored in the array. The module address code may be changed by the CPU program by using one of the control codes.



BUS INTERFACE ELECTRONICS
FIGURE 7-5.

Properly addressed control codes are decoded by the BIE section logic and the proper operation started. The control codes are:

1. Read Data Word
   The contents of the memory location specified by the address on the bus address lines are read into the BIE buffer register. The most significant 16 bits are placed on the bus data out lines followed 100 nanoseconds later by the least significant 16 bits.

2. Write Full Word
   The two 16 bit bytes are read from the bus data in lines into the buffer register to form a 32 bit word. This word is written into memory at the location specified by the address on the bus address lines.

3. Write Left Half Word
   The bus data in lines are read into the most significant bit positions of the buffer register. These 16 bits are then read into the most significant half of the memory location specified by the address on the bus address lines.

4. Write Right Half Word
   This is the same as Write Left Half Word except that the least significant bit positions of the buffer register and memory location are affected.

5. Mode Command Present
   The five most significant bits of the bus data in lines are read into the TAC section buffer register associated with this bus.

6. Read Module Mode
   The contents of the module mode register in the TAC section are read into the BIE buffer register and placed on the bus data out lines.

7. Restore Module Address
   The BIE section accesses the memory location holding the module address code and places that code into the module address register.

The operation of the module, due to the above control codes, can be altered by the module mode. This will be discussed in the mode command paragraph.

7.3.3.3.1.2 Module Address Determination. When power is initially applied to the computer, the memory address register (MAR) of each memory module can either be forced to a given state or allowed to settle into any arbitrary state. Good system design demands that all modules be identical so the initial module addresses would either all be the same or be various unknown and most likely non-repeatable codes. Either situation is

undesirable. Further, during operation it is possible that noise or other undetected system perturbations may cause one or more bits of the MAR to change giving an unknown address to the module. Some means must be provided to enable the CPU to sample or set the contents of the MAR of each memory module.

The assignment of a fixed address code (actually the four most significant bits of the 18 bit address code) that is unique to each module will do the job. The maximum number of memory modules on a memory bus will be twelve. To have the fixed codes and the MAR codes all different will require 24 codes and the addition of another bit to the address code for a total of 19 bits. This extra bit and the circuitry required to use it can be avoided if fixed and MAR codes are allowed to be idential but not for any one module and the control codes honored by the module restricted according to the way the address code is defined at the module.

If the four bit code is held in the MAR, the control code described in 7.3.3.3.1.1 will be honored by the module. If the four bit code is the fixed code, the following control codes will be honored:

1. Read Module Address

   The contents of the module address register of the BIE section are read into the buffer register and placed on the bus data out lines.

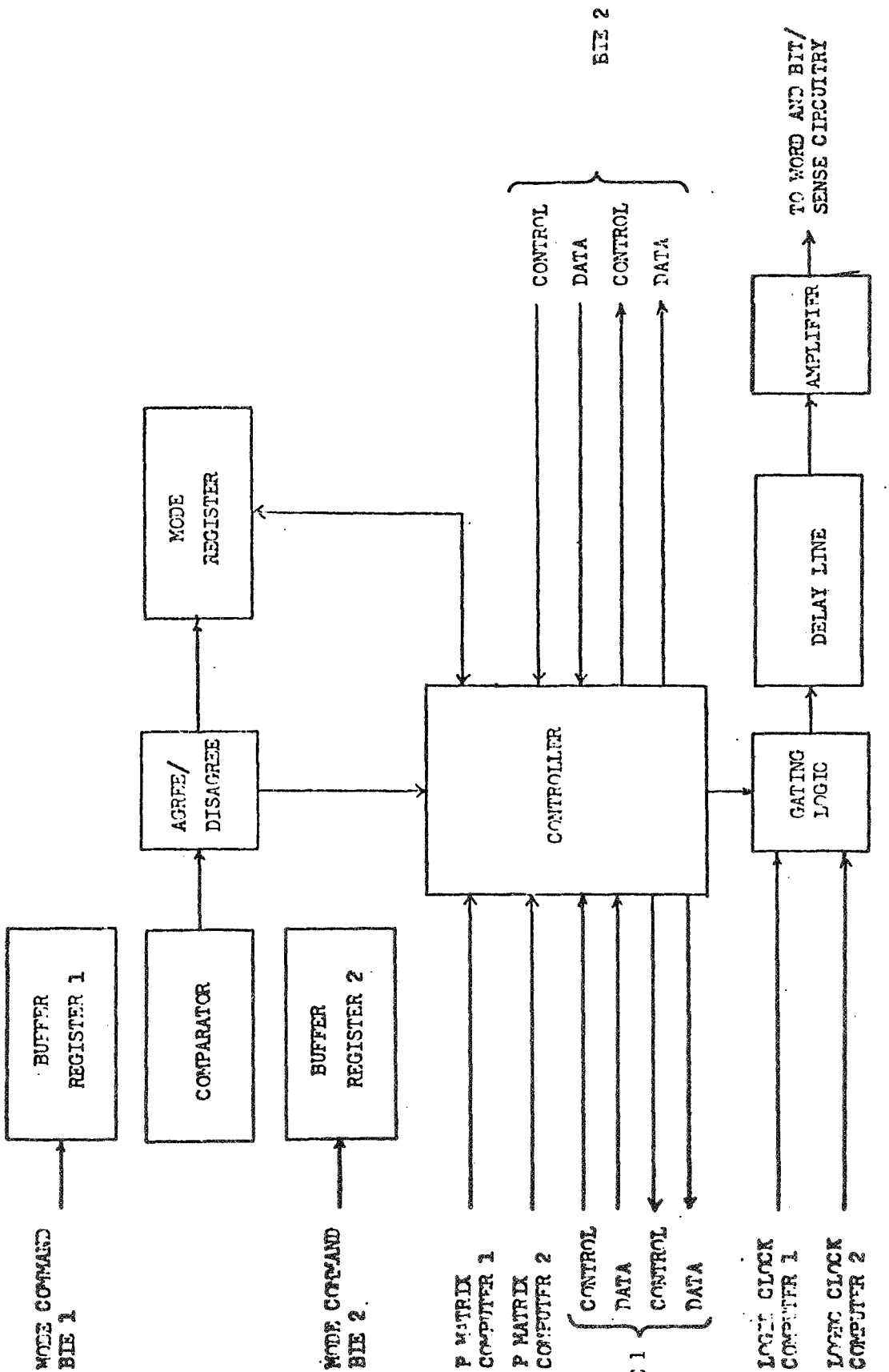
2. Store Module Address

   The BIE section reads the bus data in lines into the buffer register. The contents of the buffer register are then placed in the dedicated memory location. This memory location is then read out and placed in the module address register.

The fixed code will be set into the modules by manually attaching a plug to a computer connector. The plug will have four codes wired into it for the maximum set of computer memory modules. Each computer will have a plug which will be generated in sets of three. Each compartment which can contain three computers is considered to be independent since memory buses are restricted to within a compartment so the plugs can be identical for each compartment.

7.3.3.3.2 Timing and Control Section. The TAC section controls the memory operating modes and determines the timing for the module. A block diagram is shown in Figure 7-6.

7.3.3.3.2.1 Timing. The module timing is based upon the logic clock used in the rest of the computer. A tapped delay line is used to generate the various intermediate time intervals for proper module operation. When the computer fails (as indicated by the P matrix of the VCS), the module automatically switches to the logic clock of the other computer in the compartment. The buses operate on a request/acknowledge basis so that it is not necessary to switch between clocks during normal operation.



TIMING AND CONTROL SECTION
FIGURE 7-6

Normally, requests over the three buses are handled on a first come, first served basis. Should all buses request service at the same time, the request from bus 3 will not be serviced until after the requests from bus 1 and 2 are honored. Between buses 1 and 2, the request from the bus that was not serviced last will be honored first.

7.3.3.3.2.2 Mode Control. The operating mode of the module is determined by mode commands received from the CPU. A mode change is made only if identical mode commands are received over buses 1 and 2. Mode commands over bus 3 will not be honored by the module. Two mode commands are sent by the CPU. One command determines the operating status of buses 1 and 2. The other command determines the operating status of bus 3.

The mode commands for buses 1 and 2 are:

1. Full operation over bus 1 and bus 2:
   The module will accept and honor all control codes and all mode commands received over both buses. Priority is as described in paragraph 7.3.3.3.2.1.

2. Full operation over bus 1; no operation over bus 2:
   The module will accept and honor all mode commands received over both buses. The module will accept and honor all control codes received over bus 1. The module will not accept or honor any control codes sent over bus 2.

3. Full operation over bus 2; no operation over bus 1:
   Same as 2 except that the roles of bus 1 and bus 2 are reversed.

4. Full operation over bus 1; read only operation over bus 2:
   The module will accept and honor all mode commands received over both buses. The module will accept and honor all control codes received over bus 1. The module will accept and honor only read control codes over bus 2.

5. Full operation over bus 2; read only operation over bus 1:
   Same as 4 except that the roles of bus 1 and bus 2 are reversed.

6. Full operation over bus 1; scratch pad operation over bus 2:
   The module will accept and honor all mode commands received over both buses. The module will accept and honor all control codes received over bus 1. The module will accept and honor all control codes received over bus 2 that access a given set of locations in memory or that call for a read module address or mode operation.

7. Full operation over bus 2; scratch pad operation over bus 1:
Same as 6 except that the roles of bus 1 and bus 2 are reversed.

8. Full operation over bus 1; scratch pad and read only operation over bus 2:
The module will accept and honor all mode commands received over both buses. The module will accept and honor all control codes received over bus 1. The module will accept and honor all control codes received over bus 2 that access a given set of memory locations or that call for a read operation.

9. Full operation over bus 2; scratch pad and read only operation over bus 1:
Same as 8 except that the roles of bus 1 and bus 2 are reversed.

10. Restore Module Mode
The module will access the memory location holding the module mode command and read the contents of that location into the mode register.

The mode commands for bus 3 are:

1. No operation over bus 3:
No control codes received over bus 3 will be honored.

2. Full operation over bus 3:
All control codes received over bus 3 will be honored.

3. Read only operation over bus 3:
The module will accept and honor only read control codes over bus 3.

4. Scratch pad operation over bus 3:
The module will accept and honor all control codes received over bus 3 that access a given set of locations in memory or that call for a read module address operation.

5. Scratch pad and read only operation over bus 3:
The module will accept and honor all control codes received over bus 3 that access a given set of memory locations or that call for a read operation.

The arrival of a mode command over bus 1 or 2 starts a mode change sequence in the TAC section. The mode command is transferred into a buffer register in the TAC section that is associated with the bus over which the mode command was received. When a mode command is received over the other bus, both buffer registers are compared. If the two registers disagree, the mode register is not changed and the module continues to operate in the previous mode.

If the two registers agree, the contents of the buffer register associated with bus 1 are read into the memory location dedicated to holding the module mode command. That memory location is accessed and the contents read into the mode register.

The failure of either computer as indicated by the P matrix of the VCS of that computer will cause the TAC section to accept and honor mode commands from the remaining good computer without going through the mode command comparison sequence. When both computers are failed as indicated by the P matrices, the mode register is set to full operation over both buses and the mode command comparison logic is disabled.

7.3.3.4 Data Protection. To insure proper operation of the system, the data stored in memory may have to be protected in two ways. One type of protection is to keep bad data from being sent from the memory module and the other type is to keep certain data in memory from being changed by the CPU or IOP writing in those locations. The first type of protection can be achieved simply by generating and storing a parity bit with each half-word and checking this parity when the half-word is read out. The second type of protection can be achieved by preventing write operations into specified memory locations when the system is on line.

7.3.3.4.1 Memory Write Protect. Protection against writing in specified memory locations can be achieved in two ways although functionally, the results are the same. An attempt to write in a memory location that is protected will result in an interrupt being generated and the write operation being terminated. Selection of the protection method mechanization requires knowledge of system facts that are not now available but will be known at the time of actual hardware design. The two types of protection will be described here but no recommendation is made for selection.

7.3.3.4.2.1 Word Protect. One method is to protect each individual word in memory by storing a protect bit along with the word in memory. When a write operation is called for, the contents of the memory location are read out and the protect bit investigated. If the bit indicates protect, an interrupt is generated and the write operation terminated. If the bit indicates write, the write operation is completed.

This method is very flexible and can be used anywhere in memory without having to structure the program. The additional bit stored in memory requires a larger array (about 2.8% increase) along with one more each of a read and write channel circuitry. Perhaps the most important aspect of this method is the requirement for a read operation before a write operation. This means that a write operation will take a minimum of 1.75 microseconds instead of a normal one microsecond.

7.3.3.4.2.2 Block Protect. This method involves protecting specified blocks of memory locations by comparing the address of a write operation against protected addresses. If the addresses match, an interrupt is generated and the operation is terminated. If the addresses do not match, the operation is completed.

The addresses specifying the upper and lower bounds of each protected block must be held in logic of the memory module. The more blocks, the more addresses and also the more bits in each address. In a simple case, the protection of half the module would require the comparison of only one bit, the more significant one of the address. If half the memory is to be protected but in two separate blocks, comparison of at least the two most significant bits must be made. Thus, it can be seen that a large number of small blocks leads to large amounts of logic.

The codes specifying the block limits could be entered by means of wired plugs attached to external computer connectors. Again, the amount of hardware needed is related to the size and number of protected blocks.

Time to accomplish the write operation in this method does not increase regardless of the size or number of protected blocks since all comparisons can be done in parallel. The effective memory write cycle time will remain at one microsecond.

7.3.3.5 Memory Operations

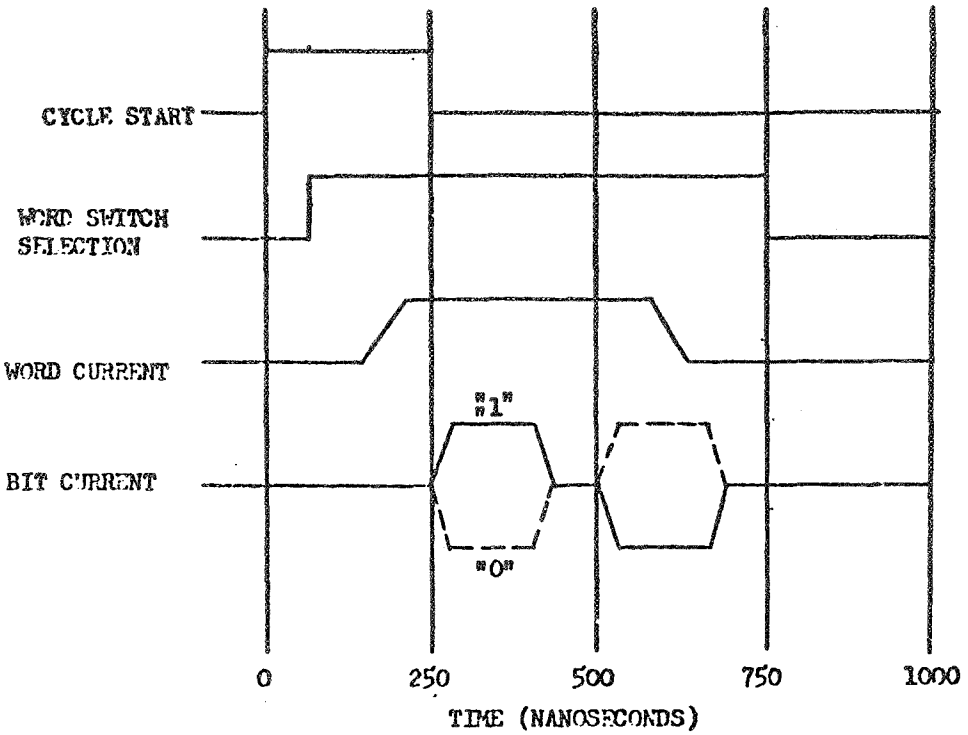
7.3.3.5.1 Write Cycle. The write operation is accomplished by the coincidence of word and bit currents at each bit address for the selected word. The polarity of the bit current establishes the storage state for the corresponding bit. A pre-write of the storage state complement is also included in the cycle to optimize wire performance. Hence, the write operation requires one word current pulse, which is coincident with the bit currents (see Figure 7-7 for timing diagram). The total memory write cycle is accomplished in less than 1.0 microsecond.

The cycle begins when the memory module receives an address and a write control word. Whether a half word or full word is to be written is determined by the control word.

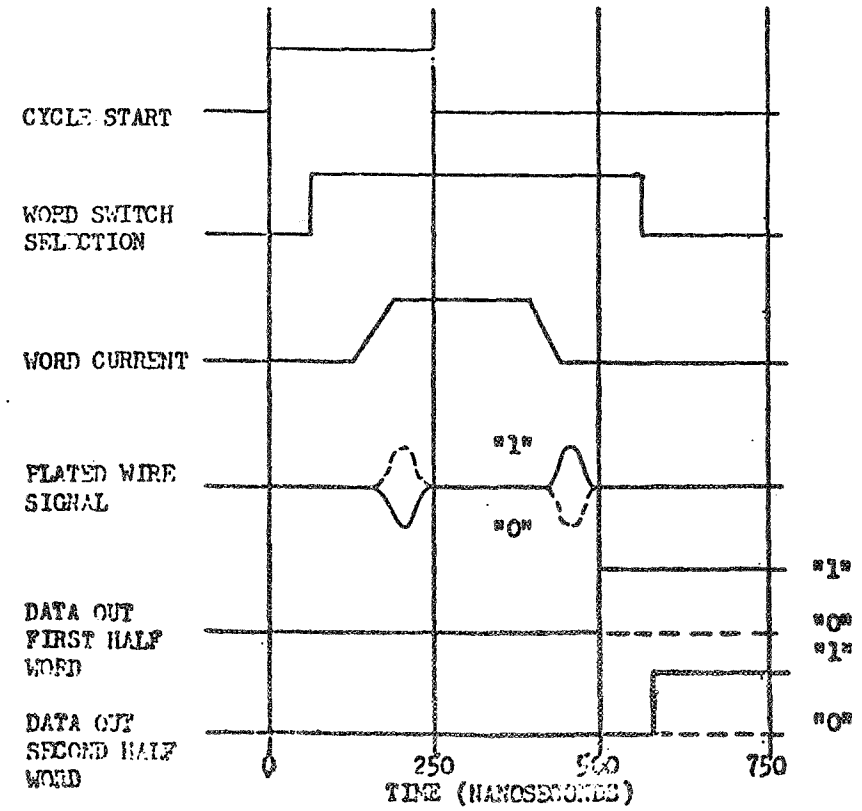
The address inputs are decoded and internally generated timing pulses command the selection of the appropriate X and Y word current switches and the proper pair of bit current switches for each bit. The word current source and bit current switches are then keyed to produce the required drive currents in the selected word and bit lines. Input data determines the sequence of the bipolar bit currents and, therefore, the polarity of each bit written.

7.3.3.5.2 Read Cycle. Readout is accomplished by applying a current down the selected word access lines and then sensing the polarity of corresponding induced voltage on the sense/bit line (plated wire) for each of the bits in the requested word. Since signal outputs will be present on all plated wires along the selected word line, the gating of the appropriate signal inputs to the sense amplifiers is made for each bit.

The cycle (see Figure 7-8 for timing diagram) begins when the memory receives the address, and a read control word. The address is decoded and internally generated timing pulses command the selection of the appropriate X and Y word current switches and gate "on" the proper MOS switch and preamplifier in the sense amplifiers for each bit. Word current is then routed



BASIC WRITE TIMING
FIGURE 7-7.



BASIC READ TIMING
FIGURE 7-8.

down the proper line, and the selected plated wire output signals are then amplified and discriminated to become available in the output data register. Information becomes available at the memory interface in 16 bit segments, the first segment 500 nanoseconds after the cycle start command and the second 600 nanoseconds after the cycle start command. A complete memory read cycle requires 750 nanoseconds.

7.3.4 Input/Output Processor Module

In Section 4.5 a description of the IOP and VCS system operation was given; in addition, a detailed description of the VCS was presented. The mechanization of the IOP will be described in this section with the VCS included in the IOP.

The Input/Output Processor (IOP) of the computer performs these functions:

a. Communication with devices external to the computer.

b. Program time synchronization.

c. Computer fault detection.

d. Voting comparison on data transmitted to local processors.

The first two functions are performed by the IOP executing commands that are stored in the computer memory or by the IOP responding to control words received from another computer. The fault detection function is built into the hardware and requires no stored commands. The last function is performed by the VCS section of the IOP hardware as specified by the setting of two matrices in the VCS section.

A block diagram of the IOP is shown in Figure 7-9.

The IOP interfaces with external devices over three types of data buses:

a. Type 1 bus for computer-to-computer communication;

b. Type 2 bus for computer-to-local processor communication;

c. Type 3 bus for computer-to-mass memory system communication.

The type 1 and 2 buses are serial channels with the messages being handled in a bit serial and word serial fashion. These messages are made up of a control word (32 bits long) and from one to 63 data words (16 bits long). All words sent over the buses will have a parity bit for each 16 bits. The parity bit is used to make each word have an odd number of binary ones in the word. The type 1 data bus consists of four (4) input channels and one output channel. The type 2 data bus consists of one input and one output channel.

The type 3 data bus is a parallel channel with the messages handled in a bit parallel and word serial fashion. The channel is bi-directional consisting of 16 data lines, one parity line and two control lines. Each word is sent over

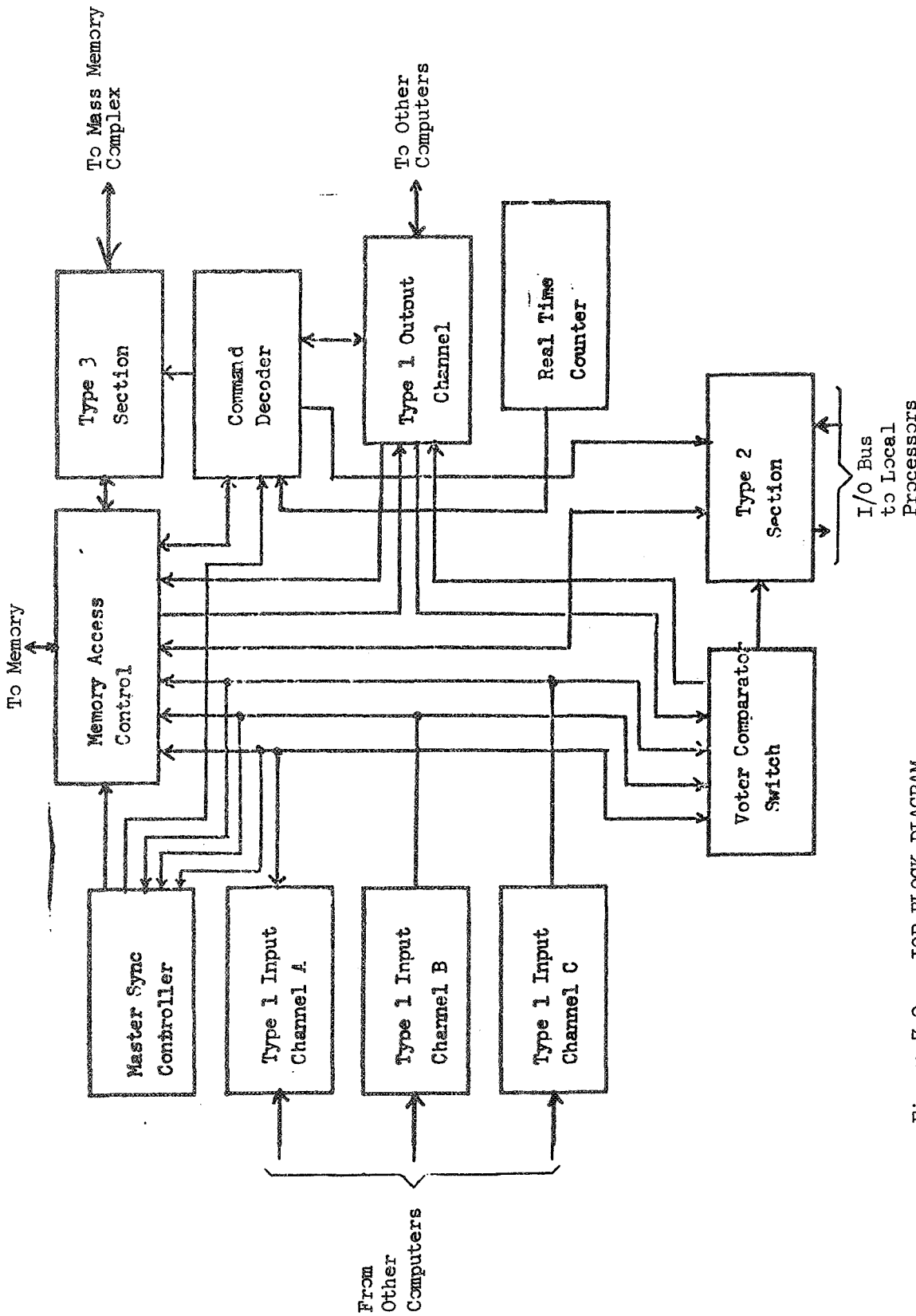


Figure 7-9. IOP BLOCK DIAGRAM

this bus on a "ready-read" basis so no timing need be furnished over the bus. Again, odd parity is used for each 16 bit word. Messages consist of a control word and from one to 127 data words. Operations over this bus are initiated by the data management system and the IOP can mask off this bus during critical data transmission modes over the other two buses.

A real time counter is included in the IOP to provide a repetitive time period reference for IOP and CPU program synchronization. The counter is counted down to zero from a set value at the logic clock rate. At zero, the counter issues an interrupt to the CPU and IOP mode controls, resets itself and starts counting down again. The length of the time period is programmable.

Detection of a computer failure is made by the built in test equipment. This consists of a counter in the IOP that operates as a "dead man" detector in that, if the counter is not reset periodically, it will reach a zero count and issue a computer No-Go discrete. This discrete is sent to all computers in the system (this is the "D" input to the VCS as shown in Figure 4-23) and, when true, causes the computer issuing it to be ignored by the operating computers. Failures both in hardware and software that prevent the resetting of this counter are detected. The counter will be reset by accessing a dedicated location in memory and loading the binary value found therein into the counter and zeroing this location after access. It is up to the CPU to reload this location to keep the counter reset.

The VCS function is included in the IOP. The operation of the VCS is controlled by the P and R matrices. The R matrix contains the mode of the computer system operation as defined by each computer. The P matrix contains the Go/No-Go status of each computer as determined by itself and other computers. The Go/No-Go status of each computer in the P matrix determines which entries in the R matrix are to be allowed to participate in system mode determination. In this way, the R matrix is adapted because of computer failures so that bad computers cannot disturb the system mode of operation. The VCS was described in detail in Section 4.5.

In order to provide data in sync ($\pm$ 1/2 word) to the VCS from the four computers, it is necessary to synchronize the four computers. The IOP contains a master sync controller that performs this function. The master sync controller receives a control word generated internally by the IOP program and also receives control words from the other computers (via Type 1 input channels). The timing of receipt of these control words determines the synchronization. This synchronization is accomplished on a periodic basis.

The IOP is set into operation by the decoding of a command and/or a control word. The commands are stored in the computer memory and are accessed according to the command program counter in the IOP. Control words are also stored in the memory and are also received from other computers in the system over the inter-computer bus. Commands are executed in sequence as any other software program. Control words are executed only when specified by a command or when a control word is received over the inter-computer bus.

The IOP command list is as follows:

1. Halt and Proceed
2. Jump
3. Conditional Skip
4. Fetch Control Word

Figure 7-10 indicates a typical layout of the IOP program. The operation of the fetch control word can be seen to mechanize the data transfer operations of the IOP. Each fetch control word instruction contains an address that points to the location of a control word. The control word contains information that describes the type of operation to be performed (e. g., input from local processors, output to another computer, etc.). Following each control word is an address that points to the location of a data block. The data block is the location into which input/output data is to be placed.

The IOP mechanization was considered in detail resulting in the detailed bit, micro sequence, and timing specifications for all the IOP functions and operations. Due to the length of the mechanization, it is reported separately in Appendix 2.

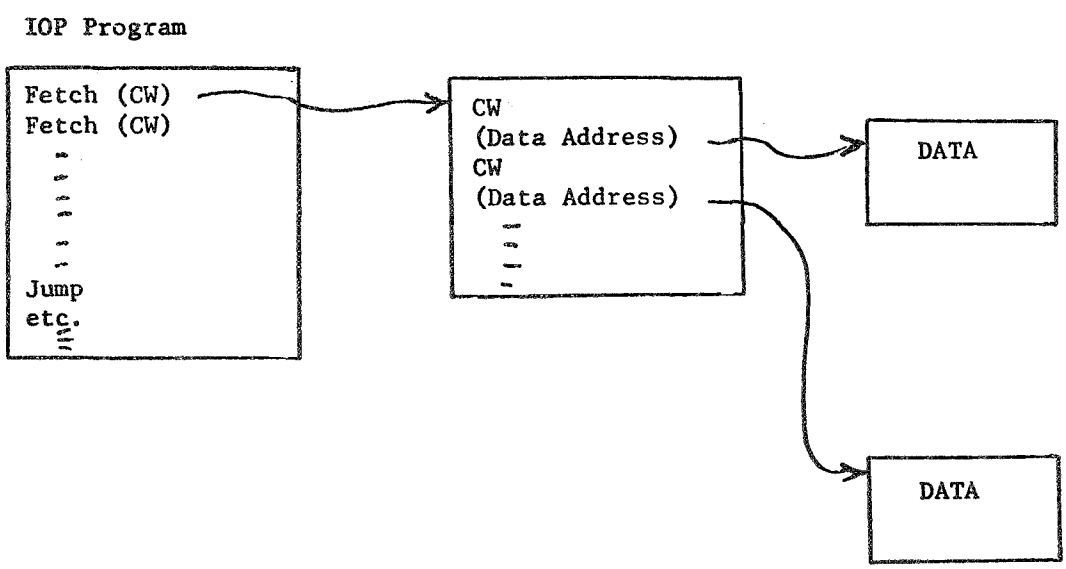


Figure 7-10. IOP Program Operation

7.3.5 Power Converter and Clock

7.3.5.1 General. The power converter and clock units supply signals that are used throughout the computer. These signals form a common reference so that the electrical levels and timing in the various areas of the computer have definite meaning with respect to each other.

7.3.5.2 Clock Unit. A master clock circuit provides the fundamental timing for the computer. This master clock consists of a temperature compensated crystal oscillator operating at a frequency high enough to satisfy the requirements of the highest speed logic in the computer. In this computer, the operating frequency of the logic will be one megahertz while the memory buses will operate at ten megahertz. Internal memory timing will be derived from the logic frequency by the use of a tapped delay line. A block diagram of the clock unit is shown in Figure 7-11.

The output of the oscillator is amplified and transmitted to the memory bus interface circuits in the CPU, IOP and memory modules. The oscillator output is also used to provide a timing signal to a four phase clock generator. The output of this generator consists of four signals that are used as timing signals in the MOS logic. These timing signals are not symmetrical and are time displaced with respect to each other but all have a period of one microsecond.

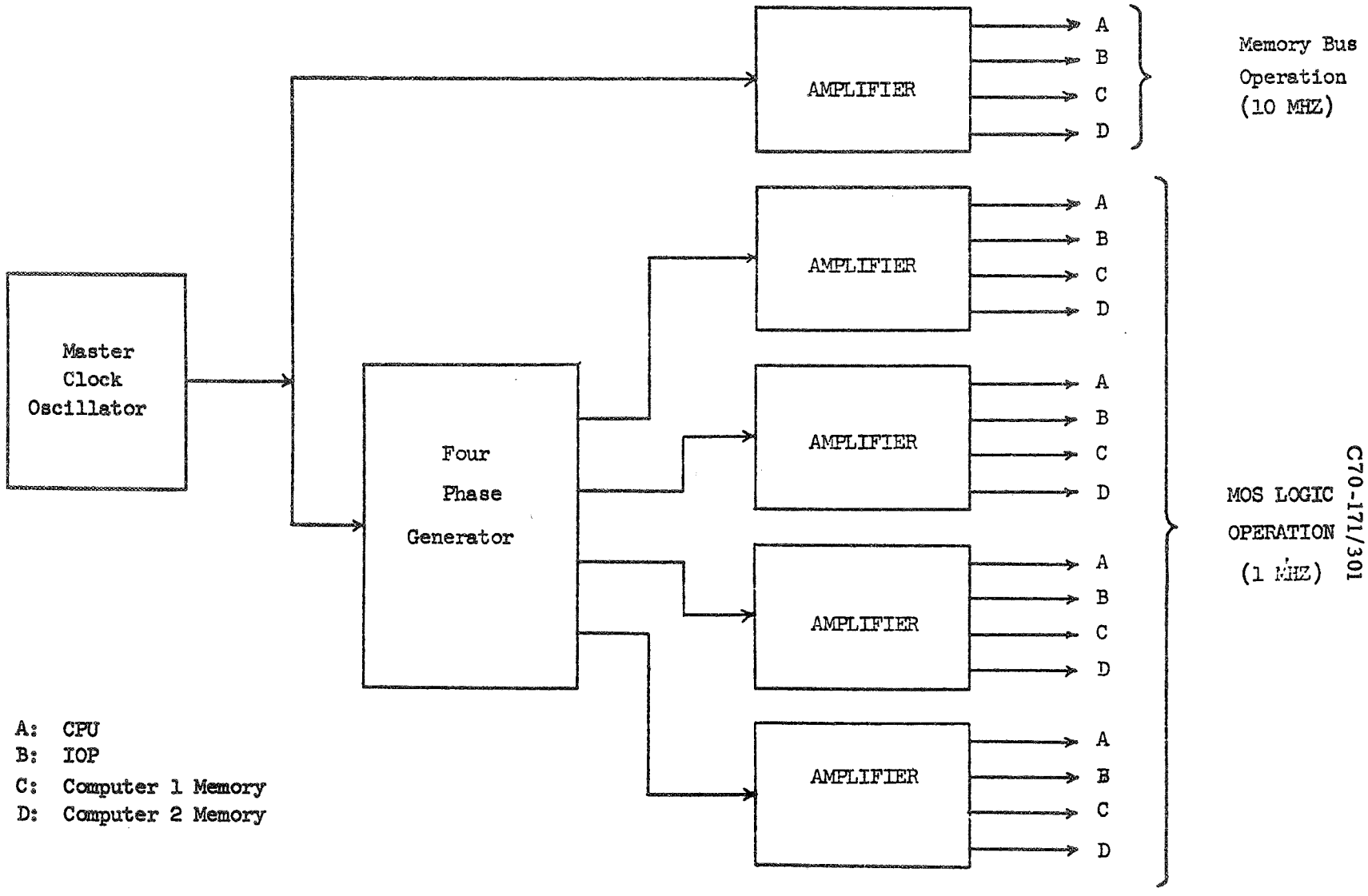
The overall accuracy of the master clock will be $\pm$ 100 parts per million for long term and $\pm$ 80 parts per million for the short term (about 48 hours).

7.3.5.3 Power Converter Unit. A power converter is necessary to match the requirements of the computer circuits with the characteristics of the primary power supply. Each computer will have a power converter capable of supplying all secondary power for the maximum computer configuration of one CPU, one IOP and four memory modules.

In Section 10 it will be shown that four independent power buses would be needed in order to meet the failure criteria of this overall system. The computer complex will be connected to these buses in such a way that the failure of two buses will result in the loss of only one computer. Table 7-2 illustrates a typical connection.

Table 7-2. Computer/Power Bus Connection

Power Bus \ Computer	I	II	III	IV	V	VI
A	X			X		X
B	X	X			X	
C		X	X			X
D			X	X	X	



CLOCK UNIT BLOCK DIAGRAM

FIGURE 7-11.

The input power is assumed to be +28 VDC with a normal working range of 24 to 28.5 volts and with transients limited from 20 to 35 volts. This will allow the use of a DC to DC type of converter with the attendant weight, size, and reliability benefits.

7.3.5.3.1 Organization. The power converter unit will have the following functional areas:

1. Load Controller
2. DC to DC Converter
3. Preregulator
4. Series Regulators
5. Switching Regulators
6. RFI Filters

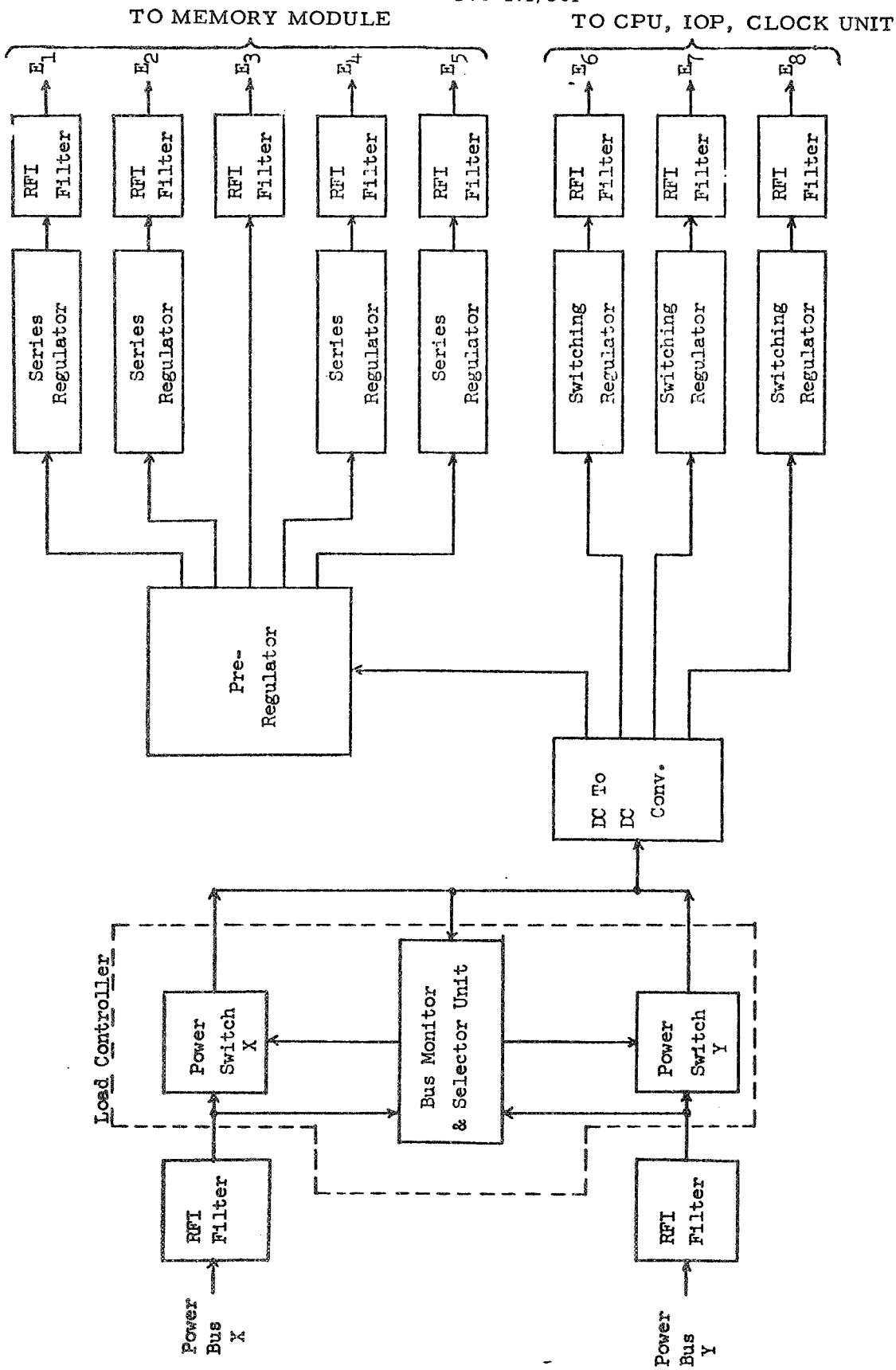
A block diagram of the power converter is shown in Figure 7-12.

7.3.5.3.1.1 Load Controller. The load controller consists of two power switches and the bus monitor and selector unit (BMSU). The load controller performs these functions.

1. Isolation - Failures in the power bus will not tend to propogate into the computer or vice versa.
2. Selection - Only one power bus will be connected to the converter input at one time.
3. Detection - Voltage level at the converter input will be monitored and a switch to the alternate power bus will be made if certain conditions are violated.

The BMSU performs function 3 and provides the control signals for function 2. The BMSU will monitor the voltage level at the input to the DC to DC converter. Voltage level excursions outside of the limits 20 to 35 volts for a period of time exceeding 50 microseconds will cause the BMSU to signal a change to the other power bus. This switching from one bus to the other will continue as long as there is sufficient power available to operate the load controller. The power for the load controller circuits will be taken from both power buses in front of the load controllers to insure operation as soon as one bus has power within the controller operating range.

The power switches perform functions 1 and 2. The switches are opened or closed due to the signals from the BMSU. The switches are connected so that one is closed when the other is open. Both switches in the same state is a failure leading to shutdown by breaking fuse links. These switches are to be solid state to reduce bus switching times and to gain increased reliability. Fast switching times result in the secondary levels of the power converter remaining constant when going from one bus to another so that computer operation is not affected. (See Section 10 for details concerning trade offs leading to mechanization selection.)



POWER CONVERTER BLOCK DIAGRAM

FIGURE 7-12.

7.3.5.3.1.2 DC to DC Converter. This unit changes the input DC power into a square wave which is applied to a transformer. This is accomplished by driving two power transistors with timing signals from a free running multivibrator. The transistors are alternately turned on and off. When On, the transistors apply 28 VDC power to either side of a center tapped primary winding of the transformer. The center-tap is connected to ground. This has the effect of producing a square wave that can be processed by the transformer. The multivibrator operates in a frequency range of 12 to 18 kilohertz.

The required voltage levels to drive the regulators are developed from the secondary windings of the transformer. These voltages are rectified, filtered and distributed to the regulators.

7.3.5.3.1.3 Pre-Regulator. This unit delivers the power to drive the logic circuits of the memory modules and the coarse regulation for the power for the memory read and write circuits. This is done to reduce the design requirements in the series regulators.

7.3.5.3.1.4 Series Regulators. These regulators are the usual type of regulator that uses a series dissipative element to control the output levels. These regulators provide the final regulation and fast response time required by the memory. In addition, one of these levels is temperature compensated to allow the memory to properly function over large temperature variations.

7.3.5.3.1.5 Switching Regulators. This type of regulator uses the principle of pulse width modulation to control the output levels. Higher efficiencies can be obtained from this type regulator over the series regulator. The regulator operates at frequencies about twenty (20) kilohertz which results in high frequency harmonics of appreciable power being developed that can cause interference in other systems. These regulators are used to supply the power requirements of the CPU and IOP logic.

7.3.5.3.1.6 RFI Filters. Components of RF energy are generated by the DC to DC converter, switching regulators and rectifier switching operations that must be suppressed. The paths that this energy may take are the conductive path over the wires connecting the power converter into the system and the radiative path through the air. Filters made up of reactive elements are placed in series with every line going to or from the power converter. These filters are low pass filters having high attenuation of frequencies above 100 hertz. The power converter will be enclosed in an RF tight enclosure to intercept the radiated energy and return it to the system ground.

8.0 SOFTWARE AND SIMULATION

8.1 INTRODUCTION

The Software and Simulation task had a three-fold objective:

1. Design and develop a software simulation of the selected computer system design developed during Task 7 (Section 7). The simulation should be suitable for use as a tool for evaluation/demonstration of preliminary design concepts and techniques.

2. Design and develop the software routines necessary to operate the simulated computer system in a FOOS environment.

3. Utilize the simulation system to debug the software and system design and to demonstrate and evaluate the feasibility and functional performance of the selected computer/software system.

The Reconfigurable G&C Computer (RGC) simulation system which was developed during this task consists of two computer programs: 1. The RGC Assembly program, and 2. The RGC Computer System Simulator program. These programs are available in a form suitable for execution at NASA MSC. The simulation system is described in more detail in Section 8.2 and Appendix 4.

The software routines programmed for execution on the Simulation System are referred to as the RGC Software System and constitute a limited "operating system" for the simulated RGC computer system. This software is composed of three sub-programs: 1) Executive program, 2) Input/Output program, and 3) Resource Controller program. These programs are described in more detail in Section 8.3 and Appendix 4.

Use of the Simulation system involved four primary activities:

1. Refining and solidifying the functional design characteristics of the selected system being evaluated during the study.

2. Debugging and evaluating operation of the Simulation system itself.

3. Debugging and refining software for the selected system design.

4. Evaluating overall system performance using fault simulation.

These activities are described in Section 8.4.

8.2 SIMULATION SYSTEM

8.2.1 General

The RGC Simulation System is comprised of two separate computer programs, the RGC Assembly Program and the RGC Computer System Simulator Program. These programs are written in the FORTRAN IV language and are compatible from a language standpoint with most medium and large scale general purpose computer system; among them, the IBM S360, CDC 6600, XDS Sigma 5/7, and Univac 1108. The programs have been executed on all the above systems except the 1108.

The two programs are executed independently. The Assembly Program is used to process programs written in a symbolic assembly language for execution on the simulated RGC computer system and convert them to a format suitable for input to the Simulator program. Additionally, the Assembly Program produces a printed listing of the programs. The Simulator Program simulates functional operation of the selected RGC computer system design by "executing" the assembled RGC computer programs and providing a printed trace of CPU, IØP, and memory activity and interaction. Simulation is performed at a functional, machine-register level and does not duplicate specific logic or circuit mechanizations.

8.2.2 RGC Computer System

Volume 2, Section 7 of this report describes the detailed mechanization of the selected RGC Computer System design. The simulation of this system design was developed to represent only the functional aspects of the design in order to demonstrate/evaluate the system's conformance to functional performance requirements, specifically the FOOS requirements. Therefore, in order to make the simulation as efficient and cost-effective as possible, the system architecture that is mechanized in the Simulator program was modified from the mechanization described in Section 7. The modifications were carefully considered to insure that the functional integrity of the simulation was not degraded. The IOP instruction repertoire is the most obvious area of change. Due to the inefficiency of dealing with bit level operations in the Fortran language, it was desirable to eliminate the IOP's compressed control word format described in the mechanization of Section 7. This was accomplished by assigning the control word functions to separate instruction types in the simulated IOP and eliminating the control word concept. It should be observed that the IOP functions remain intact, however.

On the other hand, functions considered to be particularly critical or unique in the system design are simulated in a more detailed exact manner and, in fact, portions of the VCS are simulated at a logic equation level.

The remainder of Section 8.2.2 is a description of the computer system mechanized in the Simulator Program.

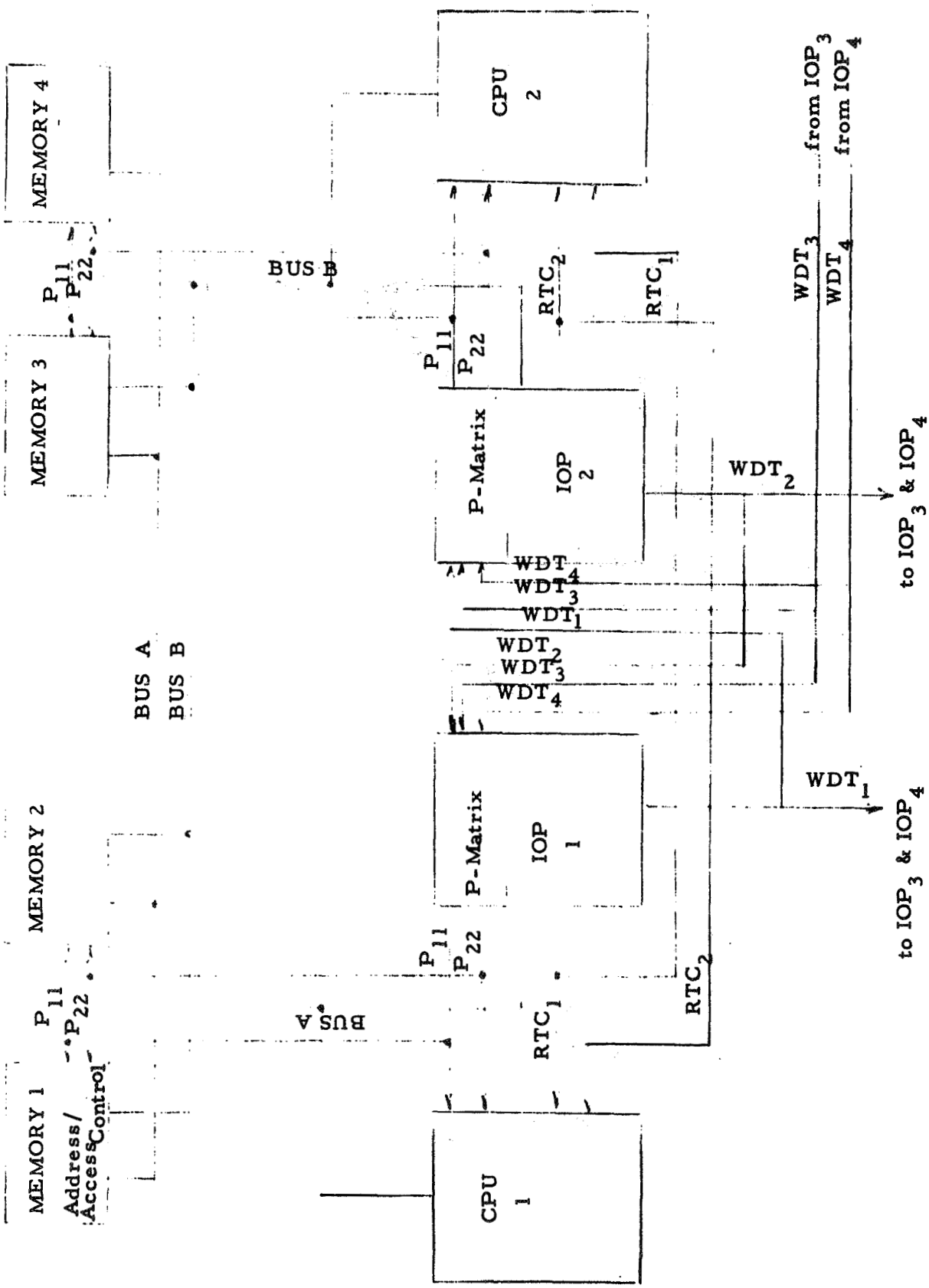


Figure 8-1. Internal Structure (Compartment)

8.2.2.1 General. The simulated computer system is comprised of the following physical modules:

4 central processing units (CPU)

4 input/output processing units (IOP)

8 memory modules

The computer system is divided into two physically separate and distinct compartments. Each compartment contains two computers where a computer consists of 1 CPU, 1 IOP, and 2 memory modules. The particular modules comprising a given computer may vary, but the term "Computer 1" always refers to the collection of modules currently associated with IOP 1. Figure 8-1 is a block diagram of compartment A which contains computers 1 and 2; compartment B is identical and contains computers 3 and 4.

The size and number of memory modules may be varied in the simulator but 8 modules with 2000 words/module will be assumed in the following description.

8.2.2.2 Memory Addressing. Each CPU can address 4 modules of main memory; i.e., all the memory in its compartment. However, the address (1, 2, 3, or 4) for a given memory module can be modified under program control; hence module 1, for instance, might represent addresses 1 to 2000 at one time and locations 6001 to 8000 another time. Moreover, the module address is unique to a memory bus and hence to a CPU/IOP combination, so each module has 2 addresses. Therefore, module 1 might simultaneously represent locations 1 to 2000 for CPU 1 and locations 4001 to 6000 for CPU 2. The SMA instruction is used to set the memory module address. Use of the instruction requires knowledge of a module's current address in order to change the address. Obviously assigning the same address to more than one module would create conflicts on the bus and make those modules inoperative.

8.2.2.3 Memory Access. Several levels of memory access control are provided to prevent inadvertent memory modification and/or accessing conflicts.

Each individual memory location contains a "storage protect" indicator which determines whether modification of its contents are allowed. This indicator is set during initial program loading. If during instruction execution an attempt is made to modify a "protected" location, the Storage Protect Interrupt (#2) is generated.

In addition, each memory module has five different access options which are selected by control commands from the two CPU's in the compartment. The access option determines what level of memory access is available to each of the two memory channels (buses).

8.2.2.3 (Continued)

The options are:

(1) Both channels open (read and write)

(2) Channel A open, Channel B closed (no read or write)

(3) Channel B open, Channel A closed

(4) Channel A open, Channel B open for read only

(5) Channel B open, Channel A open for read only.

The SAC instruction is used to select the option for each memory module. However, as long as both computers in the compartment are "good" (as determined by the diagonal elements of the P-matrix) selection of a new option can only be accomplished by agreement of the two CPU's in the compartment. When only one computer in the compartment is good, only it can select a new option. When neither computers are good, either one can change the option without agreement from the other. If during instruction execution, access is requested over a closed memory channel, the Access Violation Interrupt (#2) is generated.

8.2.2.4 Central Processing Units. Each CPU will have the following characteristics:

9 General Registers (R_1 - R_9)

1 Program Counter

100 Word Non-Alterable Local Memory (ROM)

The nine general registers may be used for address modification, as arithmetic accumulators, or as auxiliary storage registers. Register 9, however, is used as a pointer for a special data "stack" associated with each CPU, and therefore its use as a general register must be restricted.

The program counter is used to maintain the memory address of the current instruction.

8.2.2.4.1 Instruction Formats. The instruction repertoire will be described using a symbolic representation in the following general format which is compatible with the input format for the Assembly Program (Appendix 4).

ØPC SYMBØL$N, SYMBØL$N, SYMBØL$N

where:

ØPC	=	a 3-character mnemonic instruction code
SYMBØL	=	a) a decimal number b) a symbolic expression (e.g., ALPHA)
$N	=	an "$" followed by a number from 1 through 9

8. 2. 2. 4. 1 (Continued)

In general, the instruction mnemonic will be a direct computer instruction, however, a small number of "pseudo-instructions" in included as a part of the computer description whose function is to provide some action or control in the Simulator program.

There may be as many as three symbols of the form SYMBØL$N included for a given instruction where the symbol or number preceding the $ refers to a memory address and $N specifies address modification (indexing) by register N. Obviously, $N is omitted if indexing is not desired for the particular address.

8. 2. 2. 4. 2 Data Formats. Two formats are provided for arithmetic data, floating point and integer fixed point. The range of numeric data, binary format, and precision of arithmetic operations will vary depending on what computer system is being used for simulation. However, these differences should not be noticeable in programming the computer since data formats in the assembler and simulator will be decimal and all computations except integer arithmetic will be performed in floating point.

8. 2. 2. 4. 3 Indexing. Any of the general registers may be used for address modification, indexing. Normal address modification is performed; i. e., the effective operand address is determined by summing the operand address and the contents of the specified index register. Address modification is only defined when the register contains a fixed point integer.

8. 2. 2. 4. 4 Arithmetic Instructions. Both floating point and fixed point integer arithmetic is provided. Results of arithmetic operations can be accumulated in either a general register or a specified memory location. In the description of the following instructions, the term "operand address" refers to the value of SYMBØL$N (i. e., the effective address) and the term "operand" refers to the contents of the memory location or machine register designated by the operand address. Note that where the term SYMBØL$N is used to designate an operand address, any of the nine general registers may be specified at the programmer's discretion, however, indexing is not permitted in that case.

1. MV SYMBØL$N, SYMBØL$N (Move)
   The 1st operand replaces the 2nd operand.
2. MVA SYMBØL$N, SYMBØL$N, K (Move Address)
   If K = 0, the 1st operand address replaces the 2nd operand. Otherwise, the negative of the 1st operand address replaces the 2nd operand.
3. XCH SYMBØL$N, SYMBØL$N (Exchange)
   The 1st and 2nd operands are interchanged.
4. ADD SYMBØL$N, SYMBØL$N, SYMBØL$N (Add)
   The sum of the 1st operand and the 2nd operand replaces the 3rd operand. Integer fixed point addition is performed.

8.2.2.4.4 (Continued)

5. FLA SYMBØL$N, SYMBØL$N, SYMBØL$N (Floating Add)
Same as Add except floating point addition is performed.

6. SUB SYMBØL$N, SYMBØL$N, SYMBØL$N (Subtract)
The result of the subtraction of the 2nd operand from the 1st operand replaces the 3rd operand. Integer fixed point subtraction is performed.

7. FLS SYMBØL$N, SYMBØL$N, SYMBØL$N (Floating Subtract)
Same as SUB except floating point subtraction is performed.

8. CØM SYMBØL$N, SYMBØL$N (Complement)
The negative of the 1st operand replaces the 2nd operand. The operand is treated as a fixed point integer.

9. FLC SYMBØL$N, SYMBØL$N (Floating Complement)
Same as CØM except operand is treated as a floating point number.

10. MUL SYMBØL$N, SYMBØL$N, SYMBØL$N (Multiply)
The product of the two operands replaces the 3rd operand. Integer fixed point multiplication is performed.

11. FLM SYMBØL$N, SYMBØL$N, SYMBØL$N (Floating Multiply)
Same as MUL except floating point multiplication is performed.

12. DIV SYMBØL$N, SYMBØL$N, SYMBØL$N (Divide)
The result of dividing the 1st operand into the 2nd operand replaces the 3rd operand. Integer division is performed and the result is truncated.

13. FLD SYMBØL$N, SYMBØL$N, SYMBØL$N (Floating Divide)
Same as DIV except floating point division is performed.

14. FIX SYMBØL$N, SYMBØL$N (Convert to fixed point)
The 1st operand (assumed floating point) is converted to a fixed point integer (truncated) and replaces the 2nd operand.

15. FLT SYMBØL$N, SYMBØL$N (Convert to floating point)
The 1st operand (assumed a fixed point integer) is converted to floating point and replaces the 2nd operand.

16. SIN SYMBØL$N, SYMBØL$N (Sine)
The sine of the 1st operand (assumed floating point in degrees) replaces the 2nd operand.

17. COS SYMBØL$N, SYMBØL$N (Cosine)
The cosine of the 1st operand (assumed floating point in in degrees) replaces the 2nd operand.

18. SQR SYMBØL$N, SYMBØN (Square Root)
The square root of the 1st operand (assumed floating point) replaces the 2nd operand.

8.2.2.4.5 Logical Operations. All logical data will be represented as true/false (set/reset) states of a set of machine flags (flip-flops). A set of 200 of these flags will be associated with each memory module. The flags can be set, reset, tested, and further manipulated under CPU program control.

The following instructions are provided for logical manipulation. For this group of instructions the value of the operand designator FLAG$N is only defined for the range 1 to 400.

1. SET FLAG$N, FLAG$N, FLAG$N (Set flags)
   The machine flag(s) specified by the operand(s), FLAG + (R_N), are set.
2. RST FLAG$N, FLAG$N, FLAG$N (Reset flags)
   The machine flag(s) specified by the operand(s) are reset.
3. IVT I, FLAG$N, FLAG$N (Invert flags)
   The state of the block of I flag(s) specified by the 2nd operand is inverted and copied into the block starting with the 3rd operand.
4. CPY I, FLAG$N, FLAG$N (Copy flags)
   A block of I flags starting with the 3rd operand is set/reset to correspond to the state of the block starting with the 2nd operand. ($I \leq 100$).
5. AND FLAG$N, FLAG$N, FLAG$N (And flags)
   The flag specified by the 3rd operand is set/reset to correspond to the logical product of the flags specified by the 1st and 2nd operands.
6. OR FLAG$N, FLAG$N, FLAG$N (Or flags)
   The flag specified by the 3rd operand is set/reset to correspond to the logical sum of the flags specified by the 1st and 2nd operands.
7. STB I, FLAG$N (Set block)
   The block of I flags starting with the 2nd operand is set.
8. RSB I, FLAG$N (Reset block)
   The block of I flags starting with the 2nd operand is reset.
9. ANB I, FLAG$N, FLAG$N ('AND' block)
   The block of I flags starting with the 2nd operand is 'anded' (logical product) with the block starting with the 3rd operand and the result replaces the latter block.
10. ORB I, FLAG$N, FLAG$N ('OR' block)
    Same as ANB except logical sum is used instead of a logical product.

8.2.2.4.6 Data Stacking. A 50 word temporary data stack is associated with each computer. The stack locations are addressed 1 - 25 and register 9 is used as a pointer for the stack. The stack is filled from address 1 upward and the pointer always contains the address of the uppermost stack entry.

The stack is used in conjunction with BSR, RET, RRT, and IRT instructions as temporary storage for saving register data. Additionally, the stack may be used for "scratch" storage where subroutine re-entrance is desired. The following instructions are provided for that purpose:

1. MTS SYMBØL$N, K (Move To Stack)

   The first operand replaces the contents of the stack location specified by R9 + K. If K > 0, register 9 is incremented by K. (K must be an integer).

2. MFS SYMBØL$N, K (Move From Stack)

   The contents of the stack location specified by R9 + K replace the first operand. Register 9 is unchanged. (K must be an integer).

8.2.2.4.7 Branch Instruction. The following instructions are provided for conditional and unconditional modification of the program execution sequence. Note that the first operand for these instructions may not specify a general register.

1. B SYMBØL$N (Branch)

   The program counter is set to the operand address.

2. BRI SYMBØL$N (Branch Release Interrupts)

   Same as B except the interrupt level currently active is released. (See description of Interrupt System).

3. BSR SYMBØL$N, K (Branch to Subroutine)

   The program counter is set to the operand address. The previous value of registers R4, R3, R2, R1, and the program counter are stored in that order in the data stack starting at the address specified by R9 + 1 + K. Register 9 is incremented by K + 5 where $0 \leq K \leq 50$.

4. RET K (Return)

   The contents of the stack location specified by R9 replace the contents of the program counter. Register 9 is decremented by K + 5, where $0 \leq K \leq 50$.

8.2.2.4.7 (Continued)

5. RRT K (Return & Restore)

The contents of the five stack locations starting downward from the address specified by R9 replace the contents of the program counter and R1 - R4 respectively. Register 9 is decremented by K + 5 where $0 \leq K \leq 50$.

6. IRT K (Interrupt Return)

The contents of the nine stack locations starting downward from the address specified by R9 replace the contents of the program counter and R1 - R8 respectively. Register 9 is decremented by 9. In addition, the interrupt level currently active (if any) is released. (See description of Interrupt System).

7. BE SYMBØL$N, SYMBØL$N, SYMBØL$N (Branch on Equal)

If the 2nd and 3rd operands are numerically equivalent, the program counter is set to the 1st operand address. Otherwise, the program counter is incremented normally.

8. BNE SYMBØL$N, SYMBØL$N, SYMBØL$N (Branch on Not Equal)

If the 2nd and 3rd operands are numerically different, the program counter is set to the 1st operand address. Otherwise, the program counter is incremented normally.

9. BGT SYMBØL$N, SYMBØL$N, SYMBØL$N (Branch on Greater Than)

If the 2nd operand is numerically larger than the 3rd operand, the program counter is set to the 1st operand address. Otherwise, the program counter is incremented normally.

10. BLT SYMBØL$N, SYMBØL$N, SYMBØL$N (Branch on Less Than)

If the 2nd operand is numerically smaller than the 3rd operand, the program counter is set to the 1st operand address. Otherwise, the program counter is incremented normally.

11. BIL SYMBØL$N, SYMBØL$N, SYMBØL$N (Branch in Limits)

If the 2nd operand is numerically larger than or equal to the 3rd operand and smaller than or equal to the contents of the 3rd operand address plus one, the program counter is set to the 1st operand address. Otherwise, the program counter is incremented normally.

8.2.2.4.7 (Continued)

12. BØL SYMBOL$N, SYMBOL$N, SYMBOL$N (Branch Out of Limits)

If the 2nd operand is numerically smaller than the 3rd operand or larger than the contents of the 3rd operand address plus one, the program counter is set to the 1st operand address. Otherwise, the program counter is incremented normally.

13. BI SYMBØL$N, SYMBØL$N, DATA (Branch and Increment)

The sum of the 2nd operand and the value of DATA replace the 2nd operand. If the sum is negative, the program counter is set to the 1st operand address. Otherwise, the program counter is incremented normally.

14. BS SYMBØL$N, FLAG$N, FLAG$N (Branch on Set)

If the flags specified by the 2nd operand and 3rd operand are set, the program counter is set to the 1st operand address. Otherwise, the program counter is incremented normally.

15. BR SYMBØL$N, FLAG$N, FLAG$N (Branch on Reset)

If the flags specified by the 2nd operand and 3rd operand are reset, the program counter is set to the 1st operand address. Otherwise, the program counter is incremented normally.

16. BM SYMBØL$N, FLAG$N, FLAG$N (Branch on Mixed)

If the state of the flags specified by the 2nd and 3rd operands is different, the program counter is set to the 1st operand address. Otherwise, the program counter is incremented normally.

17. BMS SYMBOL$N, FLAG$N, FLAG$N (Branch on Mixed Specific)

If the flag specified by the 2nd operand is set and the flag specified by the 3rd operand is reset, the program counter is set to the 1st operand address. Otherwise, the program counter is incremented normally.

18. BBS SYMBØL$N, I, FLAG$N (Branch on Block Set)

If the block of I flags starting with the 3rd operand is set, the program counter is set to the 1st operand address. Otherwise, the program counter is incremented normally.

8.2.2.4.7 (Continued)

19. BBR SYMBØL$N, I, FLAG$N (Branch on Block Reset)

If the block of I flags starting with the 3rd operand is reset, the program counter is set to the 1st operand address. Otherwise, the program counter is incremented normally.

20. BBE SYMBØL$N, FLAG$N, FLAG$N (Branch on Block Equal)

If the block of 4 flags starting with the 2nd operand is logically equivalent to the block of 4 flags starting with the 3rd operand, the program counter is set to the 1st operand address. Otherwise, the program counter is incremented normally.

8.2.2.4.8 Control Instructions. The following instructions provide basic mode control:

1. WI (Wait for Interrupt)

The program counter is incremented by one and CPU enters a "non-compute" mode. The "compute" mode will be reentered when any interrupt is received, at which time normal execution will be resumed.

2. HLT (Halt)

The CPU enters a "halt" mode. No interrupts are recognized in this mode.

3. SMA SYMBØL$N, SYMBØL$N (Select Module Address)

The address of the memory module specified by the 1st operand address is set to the 2nd operand address.

4. SAC SYMBØL$N, SYMBØL$N (Set Access Control)

This instruction selects the memory access option specified by the 2nd operand address for the memory module specified by the 1st operand address. The access options are as follows:

(1) Both channels open
(2) Channel A open, channel B closed
(3) Channel B open, channel A closed
(4) Channel A open, channel B open for read only
(5) Channel B open, channel A open for read only

8.2.2.4.9 Read Only Memory Module (ROM). Each CPU contains a 100 word non-alterable memory. The primary purpose of this memory is to provide capability for autonomous CPU operation to assist in module level reconfiguration. This memory is addressed as locations 1 to 100 and two instructions are provided to enable program branching between main memory and the ROM.

1. BLM SYMBØL$N (Branch to Local Memory)

   The local memory mode is established. The program counter is set to the 1st operand address (in the ROM).

2. BMM SYMBØL$N (Branch to Main Memory)

   The main memory mode is established. The program counter is set to the 1st operand address (in main memory).

8.2.2.4.10 Interrupt System. Six interrupt levels are provided for each CPU. A different priority is associated with each level. Two levels of control are provided for the interrupt system: 1) the entire system can be enabled or disabled under CPU control, and 2) each individual interrupt level can be armed or disarmed selectively under CPU control. When an individual interrupt level is disarmed, no interrupt at that level is recognized r retained. When a level is armed, the interrupt level will become active if: 1) an interrupt is present, 2) the interrupt system is enabled, and 3) no level of higher priority is currently active.

When an interrupt level becomes active, the CPU interrupts the program sequence, stores the contents of registers, R8, R7, ..., R1, and the program counter in the data stack advancing R9 by 9, and executes the instruction in one of six fixed locations in the ROM. This instruction will normally be a BMM or B instruction which transfers control to the interrupt processing subroutine. Normally an interrupt level, once active, remains in that state until an interrupt return instruction (IRT or BRI) is executed signifying that processing has been completed. However, the level may be temporarily made inactive if a level of higher priority requires servicing.

The following instructions are provided for control of the interrupt system. It should be noted that an interrupt level will not become active until after execution of one instruction subsequent to an EI instruction.

1. EI (Enable Interrupts)
   The interrupt system is enabled.

2. DI (Disable Interrupts)
   The interrupt system is disable.

3. AI I, I, I (Arm Interrupts)
   The interrupt(s) specified by the operand(s) are armed (I = 1, 2, ..., 6)

8.2.2.4.10 (Continued)

4. DAI I, I, I (Disarm Interrupts)
The interrupt(s) specified by the operand(s) are disarmed. (I = 1, 2, ..., 6)

5. EXI I, I, I (Execute Interrupts)
The interrupt(s) specified by the operand(s) are initiated. (I = 1, 2, . . ., 6)

The interrupt assignment is as follows where interrupt 1 is the highest priority:

1. Power Up
2. Storage Protect/Access Violation
3. P_{11} or P_{33} (Z_A or Z_C)
4. P_{22} or P_{44} (Z_B or Z_D)
5. Primary RTC*
6. Secondary RTC*

8.2.2.4.11 Pseudo Instructions. For convenience in the simulation software system, a set of "pseudo" instructions is included in the CPU instruction set which would normally not be available and may have no effect on the CPU itself, but which provides control of special simulator features during program execution. These instructions are assigned addresses in the object computer memory but no time is associated with their execution.

1. DLY SYMBØL$N,K (Delay)

This instruction causes the CPU to "waste" the amount of time specified by the value of the 1st operand and to replace the 1st operand with the value K. No other egisters or memory locations are affected and the timing relationship between other CPU and I/O processors is maintained.

2. P ABC... (Print)

The specified string of up to 56 alphanumeric characters is printed on the simulator output listing together with the simulated computer cumulative execution time.

*The primary RTC is the one from the corresponding IOP (RTC_1 for CPU1) The secondary RTC is the one from the other IOP in the compartment.

8.2.2.4.11 (Continued)

3. PD SYMBØL$N, SYMBØL$N, SYMBØL$N (Print Decimal Integer)

The operand(s) is printed on the simulator output listing. The operand(s) is assumed to be an integer.

4. PFD SYMBØL$N, SYMBØL$N, SYMBØL$N (Print Floating Decimal)

Same as PD except the operand(s) is assumed to be a floating point number.

5. PF I, FLAG$N (Print Flags)

The states of the block of I flags starting with the 2nd operand are printed on the output listing.

8.2.2.4.12 CPU Instruction Timing. **The following table specifies the execution time for each instruction type. No units need be associated with the times since only relative time is significant for the simulation, however for convenience they are referred to as microseconds.**

INSTRUCTION	EXECUTION TIME
ADD, FLA, SUB, FLS, BIL, BØL, CPY, ANB, ØRB, AND, ØR	3
MUL, FLM, SIN, COS, SQR	5.5
DIV, FLD	10
SCH	3.5
MVA, SET, RST, STB, IVT, RSB, BS, BR, BM, BBS, BI, BBR BMS, RET, MTS, MFS	2
B, EI, DI, AI, DAI, EXI, WI, HLT, BRI, SMA, SAC	1
BSR, RRT, IRT	6
BE, BNE, BGT, BLT, MV, CØM, FLC, FIX, FLT	2.5

8.2.2.5 Input/Output Processing Unit. **Each IOP is an independent processor having the following characteristics:**

Stored program control
Real time clock
Watchdog timer
Four (4) independent serial input channels
Two (2) serial output channels
Voting/comparison logic on one (1) output channel

8.2.2.5 (Continued)

Figure 8-2 is a block diagram of the data channels connecting the IOP with external subsystems (LPs) and with other IOPs. The voting/comparison logic has been called a VCS and is described in detail in Section 4.

8.2.2.5.1 Memory Addressing. Each IOP can address only 2 modules of main memory. From an addressing standpoint these two modules are always numbers 1 and 2. However, as previously indicated, the CPUs can reassign the memory module addresses; hence, each IOP may actually operate with any memory module, in the same compartment.

8.2.2.5.2 Real Time Clock. Each IOP contains an independent timing source which is used to generate a fixed frequency clock pulse referred to as the Real Time Clock (RTC). This clock is used to maintain a real time reference in both the CPUs and the IOPs. The RTC associated with a given IOP is routed in the form of a program interrupt to the two CPUs in the corresponding computer compartment.

The RTC is normally used as a "proceed" indication in the IOP program causing the IOP to continue after execution of an IDL (Idle) instruction. If the RTC occurs in other than an idle mode the IOP program counter is set to location 7.

8.2.2.5.3 Master Sync. In order to allow for synchronous operation of the four IOPs, a sync controller is provided in each IOP which derives a "master sync" signal from the four RTCs in the system. The sync controller is initiated by an SNC (Sync) instruction and the process is as follows:

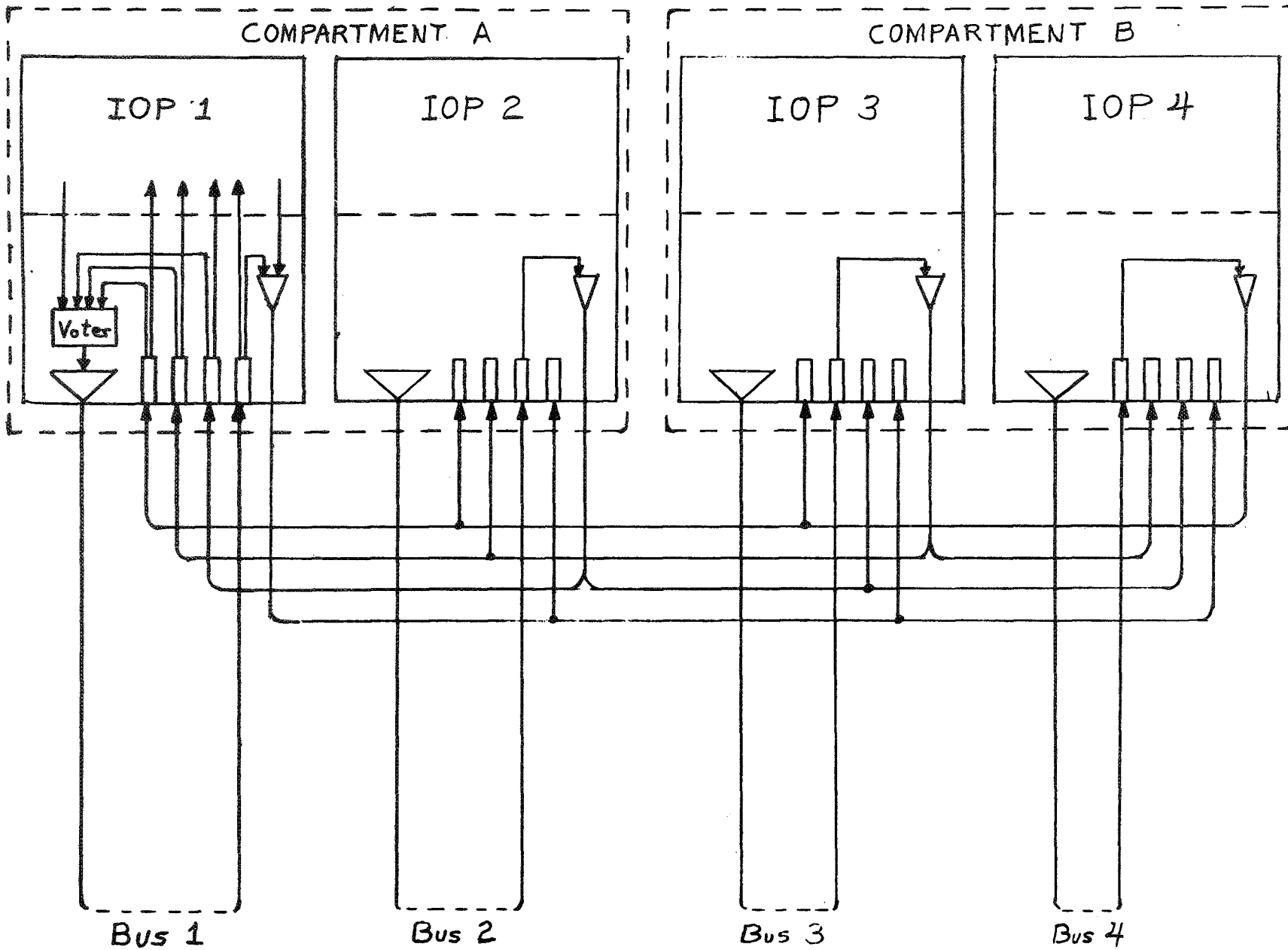
(1) The IOP transmits a sync message addressed to all computers. This message is initiated a fixed time interval prior to the RTC pulse where the interval is precisely the transmission time for the message. All operating IOPs likewise transmit a sync message.

(2) If at least two computers are included in the synchronization process as specified by the mask word of the SNC instruction, then the second sync message received is used as a "master sync" point. This causes the RTC time to be reset and the IOP program to proceed after a 6 μs delay.

(3) If only two computers are indicated in the mask and a second sync message is not received within 6 μs or if only one computer is indicated by the mask, then the receipt of the first sync message is used as the master sync point.

(4) Note that the IOPs own sync message is included if indicated in the mask word. In this case, the IOP accounts for the transmission time of the sync message and "simulates" receipt of its own transmission.

Figure 8-2. I/O Buses



C70-171/301

8.2.2.5.3 (Continued)

(5) If at the time of execution of an SNC instruction, the RTC timer is less than the sync message transmission time plus 6 μs, then the IOP program counter is set to location 7.

(6) Sync messages received at any time other than during execution of an SNC instruction are ignored.

8.2.2.5.4 Watchdog Timer. Each IOP contains a continuously running interval timer referred to as a Watchdog Timer (WDT). The timer is counted down and when it reaches zero, the IOP automatically reloads it with the contents of location 5 and stores zero into location 5. If the WDT ever goes negative, a no-go signal is sent to the P-matrices in each IOP which will reset the diagonal element corresponding to the IOP issuing the signal. The P element going false triggers an interrupt to the two CPUs in the corresponding compartment. Additionally, the IOP program counter is set to location 6. This operation occurs regardless of IOP mode.

The WDT is used to detect gross CPU malfunctions which result in loss of CPU program control. In normal operation, the CPU program would reload the WDT reset value in location 5 at regular intervals such that the WDT would not run out. The interval of the WDT timer is under CPU program control, but is limited to a maximum of approximately 60 MS

As previously indicated, the WDT from a given IOP is used to determine the state of the P-matrix diagonal element corresponding to that computer. Hence, this signal is routed to the P-matrices in every other IOP.

8.2.2.5.5 IOP Instructions.

1. OS DATA, VCS, LP (Output to Subsystem)

This instruction initiates a data transfer from the IOP to an external subsystem through a VCS (in the same IOP or another IOP). The memory location specified by DATA contains the number of words (excluding the output control word) to be transmitted and DATA + 1 contains the first word of the message. The symbol, VCS, specifies the first of two consecutive memory locations, each of which contains a VCS address (an integer, 1-4). The value of the symbol, LP, is the sum of the Local Processor address (an integer between 0-20) and the output control code 36520.

The output transmission process is as follows:

A. Data is transmitted simultaneously or individually from any/all IOPs to a given VCS. The first word transmitted is the output control word which is the sum of the output control code, 36520, the LP address, and the VCS address times 10^5. The VCS compares the data from the IOPs on a word basis where sync between IOPs must be main-

8.2.2.5.5 (Continued)

tained ± 15 μs. Selection of the comparison/voting mode in each VCS is controlled by the state of the R-matrix which is set by mode control commands from all four computers (see LR instruction). Data transmission is word and bit serial (16 bits + parity per word) at a 1 MH rate.

B. The results of the data comparison/voting process in the VCS determine what data (if any) is transmitted on the data bus to the local processor. Each bus is a closed loop, beginning and ending at a VCS. As data is transmitted, it is simultaneously received in the same VCS and routed on a common bus to all IOPs.

C. Each IOP which is transmitting data automatically compares this "feedback" of the data with the data it has transmitted on a word by word basis. When the entire message (one output instruction) has been transmitted, the IOP automatically transmits one more word, a "Go/No-Go" indicator. This word is taken from one of two dedicated memory locations, depending on the results of the feedback comparison. Location 8 is the Go indicator and 9 is the No-Go.

D. The feedback of the Go/No-Go word is examined in the IOP. If "Go" is indicated, the IOP proceeds to the next program instruction. If "No-Go," the entire message (hence the OS instruction) is repeated. If the second transmission is also No-Go, a third is attempted. If the third fails, the second of the two VCS address words is selected and the transmission is attempted up to three (3) more times. If all six (6) transmissions are faulty, the IOP reselects the first VCS address word and proceeds to the next program instruction. If transmission is successful using the second address word, this selection is retained for subsequent I/O operations until the occurrence of three successive faulty transmissions which cause a return to the first address.

E. If, during the output process, the feedback does not occur or is interrupted (due to no majority at the VCS or to noise on the bus) the transmission is terminated and the retransmission process initiated.

8.2.2.5.5 (Continued)

2. OSN DATA, VCS, LP (Output to Subsystem, No Retransmission)

Same as OS except that no attempt is made to repeat the transmission when a No-Go is indicated. Note that the symbol VCS specifies only a single VCS address word for this instruction.

3. IS DATA, VCS, LP (Input from Subsystem)

This instruction initiates the transmission of data from a subsystem to the computer system. The memory location specified by DATA contains the number of words to be transmitted and the data words are to be stored starting at DATA +1. The symbol VCS specifies the first of two memory locations which contain codes designating which data bus(es) are to be used for the reply transmission. The code is an integer ranging from 1 to 4321 and is comprised of from one to four of the integers 1, 2, 3, or 4 where each integer can appear only once.

Examples:

123	Transmit data request over bus 1 and receive over buses 1, 2, and 3.
2431	Transmit over 2; receive over 1, 2, 3 and 4.

The value of the symbol, LP, is the sum of the local processor address (0-20) and the input control code 41630. The input process is as follows:

A. An input request is transmitted simultaneously or individually from any/all IOPs to a VCS. The input request consists of two words. The first word is the sum of the input control code 41630, the LP address, and the reply bus code times 10^5. The second request word is the message length (DATA +1). The "output" of the input request words through the VCS and back to the participating computers is identical to the data output process previously described except that the Go/No-Go indicator is not transmitted. The feedback comparison is performed however.

B. After the input request is transmitted, the IOPs await an "acknowledge" message from the LP transmitted on the same bus as the request. The acknowledge message is the sum of the first input request word and the constant 50, and is compared with the expected response by all participating IOPs.

8.2.2.5.5 (Continued)

C. Retransmission of the input request is attempted if either of the following conditions occur.

(1) The feedback of the input request fails to compare.
(2) The acknowledge is not received within 84 μs or is incorrect.

The retransmission and VCS address switching process is the same as described for output transmissions except that the VCS address selection affects the selection of input buses as well as the output bus. However, if only the acknowledge message is at fault, no VCS address switching occurs.

D. If no retransmission is indicated, the data input process will proceed and data transmitted from the LP will be stored in memory. At the conclusion of the message the IOP stores one more word which is equal to 1 if no errors were detected and -1 if transmission was unsuccessful or faulty.

E. If retransmission is required, storing of data resulting from a faulty LP transmission is inhibited.

4. ISN DATA, VCS, LP (Input from Subsystem, No Retransmit)

Same as IS except no retransmission is attempted. Note that the symbol VCS specifies only a single VCS address word for this instruction.

5. OC DATA, COMP, ADDR. (Output to Computer)

This instruction initiates a data transmission to one or more of the other computers (IOPs). The memory location specified by DATA contains the number of words to be transmitted and DATA +1 contains the first word of the message. The IOP(s) being addressed is determined by a code contained in the location specified by COMP. The code contains a decimal integer between 1 and 4321 and is comprised of the integers 1, 2, 3 and 4, each being used only once.

Example: 123, 321, etc. means transmit to computers 1, 2 and 3.

The location specified by ADDR. contains an integer, 0 - 9. This integer is used to select one of 10 locations in the receiving computer which will contain the address where the incoming data is to be stored. A separate block of ten (10) locations is dedicated to each IOP input channel (hence, to every other IOP). The memory assignment is as follows:

8.2.2.5.5 (Continued)

IOP 1	location 10 - 19
IOP 2	location 20 - 29
IOP 3	location 30 - 39
IOP 4	location 40 - 49

For example, assume that an OC instruction is being executed by IOP 2, location COMP contains 3, and location ADDR contains 5. The instruction would cause data to be transferred from IOP 2's memory starting with DATA +1 to IOP 3's memory starting with the address contained in location 25.

Data transmission is bit and word serial at a 1 MH rate where a word is 16 bits plus parity. No acknowledge is given by the receiving IOP(s) and no feedback comparison is performed on the transmitted data. (See General Note.)

6. LR DATA, VCS (Load R)

This instruction initiates transmission of a mode control message to one or more VCSs. The four machine flags starting with the one specified by DATA are used to set the appropriate row in the R matrix of the VCS(s) determined by a code in the location specified by VCS. The code is analogous to the ones described previously and allows selection of any combination of the four VCSs. (See General Note.)

7. LP DATA, VCS (Load P)

This instruction is analogous to LR and is used to set three elements of the appropriate row in the P-matrix of one or more VCSs.

8. SAR DATA, VCS (Sample R)

The VCS mode determined by the current R-Matrix configuration is stored in four consecutive machine flags starting with the one specified by DATA. The location specified by VCS contains the VCS address (1, 2, 3 or 4). (See General Note.)

9. SAP DATA, VCS (Sample P)

This instruction is analogous to SR and stores the four diagonal elements of the P-matrix. (See General Note.)

10. SAS DATA, VCS (Sample S)

This instruction is analogous to SR and SP and stores the contents of the appropriate row of the S matrix. (See General Note.)

8.2.2.5.5 (Continued)

11. RS VCS (Reset S)

This instruction sets the elements of the appropriate row in the S matrix to zero. The location specified by VCS contains the VCS address (1, 2, 3 or 4). (See General Note.)

12. J SYMBOL (Jump)

The program counter is set to the value of SYMBOL.

13. BC SYMBOL, COUNT, INIT (Branch and Count)

The contents of the location specified by COUNT are incremented by one. If the incremented value exceeds 31, the contents of COUNT are set to the value of INIT and the program counter is set to the value of SYMBOL. Otherwise the program counter is incremented normally.

14. TF SYMBOL, FLAG (Test Flag)

If the machine flag specified by FLAG is set (1), the program counter is set to the value of SYMBOL. Otherwise the program counter is incremented normally.

15. SF FLAG (Set Flag)

The machine flag specified by FLAG is set.

16. RF FLAG (Reset Flag)

The machine flag specified by FLAG is reset.

17. IDL (Idle)

The IOP enters an idle (non compute) mode during which no further instructions are executed. This mode is retained until a "proceed" signal occurs due to the real time clock. When this occurs, the IOP proceeds to the next instruction in sequence.

18. SNC MASK (Sync)

The IOP enters an idle (non-compute) mode during which no further instructions are executed. This mode is retained until a "Start Cycle" signal is generated by the master sync controller (see description of master sync). When this occurs, the IOP proceeds to the next instruction in sequence. The four consecutive flags starting with MASK specify which IOPs are to participate in the sync process.

8.2.2.5.5 (Continued)

GENERAL NOTE: (OC, LR, LP, SR, SP, SS, SAM, RS, OS, OSN, IS, ISN)

As pictured in Figure 8-2, output data from a given IOP is transmitted on the same bus as is used for feedback data or input data from the system bus associated with that IOP. Therefore, IOP 4 cannot send data to another IOP while Bus 4 is being used. To resolve a potential conflict, data on the system bus is given priority. Hence, if an OC instruction, for example, is being executed in IOP 4 when a transmission is initiated on Bus 4, execution of the OC instruction will be terminated, re-initialized, and remain pending until the bus becomes available, at which time it will begin execution again. Likewise, if the OC instruction is encountered while the bus is busy, execution of the instruction will be delayed until the bus is free.

This potential delay or interruption of instruction execution can occur with any instruction requiring transmission of data to another IOP; i.e., OC, LR, LP, SR, SP, SS, SAM, RS, OS, OSN, IS, ISN. Note that no delay can occur on these instructions when they are addressed to the VCS in the same IOP.

8.2.3 RGC Assembly Program

The Assembly Program for the Reconfigurable G&C Computer System (RGC) is used to process programs written in a symbolic assembly language and prepare them for input to the RGC Computer System Simulation Program. The architecture and functional characteristics of the RGC computer system are described in Section 8.2.2. Programs written for this computer system are input to the Assembly program in the form of a punched card deck. The Assembly program produces a printed listing of the programs and a punched card deck suitable for input to the RGC Computer Simulation program (Section 8.2.4).

The Assembly Program is written in Fortran IV for execution on the IBM S360/65 at Autonetics and the XDS SIGMA 5 at NASA MSC. However, the language is compatible with the Fortran compilers for the Univac 1108 and CDC 6600 systems also.

The program is constructed in a "two-pass" organization common to many symbolic assemblers. During the initial pass over the input card deck, all symbolic references are evaluated and a table of symbol values is constructed. Some checking of input format is also performed in the first pass. As the input cards are read during pass 1, they are saved on magnetic disc storage for more rapid access during pass 2.

During the second pass, the symbol table constructed during pass 1 is used to assign memory addresses to all instructions, operand references, data items, etc. This information is punched in a compressed format (3 instructions/card) in the object deck together with control information used by the Simulation program during the loading process. Further error checking is performed in pass 2, and the results of the assembly process are printed in a one-input-card/line format on the program listing.

A user-oriented description of this program is contained in Appendix 4 of this report.

8.2.4 RGC System Simulation Program

The Simulation Program for the Reconfigurable G&C Computer System (RGC) is designed to simulate execution of programs written for the RGC system and input in card deck form in the format produced by the RGC Assembly program. The architecture and functional characteristics of the RGC computer system are described in Section 8.2.2.

The simulation can be characterized as functional and interpretive, indicating that only the functional performance of the RGC system is duplicated and that successive program instructions are individually examined and interpreted to determine appropriate simulated actions.

The Simulator program is constructed in a highly modular fashion to facilitate modification. There are six major program modules: 1. Main Executive, 2. Input Processor, 3. CPU Executive, 4. IØP Executive, 5. VCS Simulator, and 6. Fault Generator.

8.2.4 (Continued)

The Main Executive maintains overall simulation timing, sequencing, and mode control. A "ring" structure is employed to control timing and sequencing. The ring defines the sequence and duration of simulated execution of each of the IOPs, CPUs, and VCSes and initiates the Fault Generator when necessary. When conditions in the simulation indicate the necessity to update a given processor, a "call" is placed in the ring for that processor for the time at which the conditions were established. For example, executing an IOP instruction which caused simulated data transmission to a VCS would result in insertion of a call to that VCS in the ring at the time of arrival of the data. In this manner the parallel operation of the RGC system processors is realistically simulated in a necessarily serial manner.

The Input Processor is used to load the RGC programs into the simulated memory modules and to initialize the simulation.

The CPU and IOP Executives control simulated execution of the CPUs and IOPs respectively. The unique status of each processor being simulated is maintained in a common status area such that common routines are used to simulate all CPUs or IOPs. The bulk of the simulated execution is performed by the Instruction Processor module which consists of a unique subroutine for each CPU and IOP instruction type.

The VCS Simulator simulates the detailed logic in the four VCSes. The timing relationships between the VCSes and the IOPs is very critical to system operation. The time resolution in the simulation is therefore maintained more precisely when VCS activity is being simulated.

The Fault Generator performs two basic functions. Initially, it processes the data on the fault list cards which select pre-planned faults to be simulated. Subsequently during simulation of malfunctions, it interacts with the CPU and IØP Executives and the VCS Simulator to alter conditions in the simulation affected by the fault being simulated. Much of the Fault Generation is merely program "linkages" inserted in the normal execution sequence which provide for transfer of data and control between the Fault Generator and the other program routines to allow for the necessary interaction. These linkages can be readily used for insertion of additional fault types or specific malfunction conditions not included in the set of pre-planned faults.

A user-oriented description of this program is contained in Appendix 4 of this report.

8.3 RGC SOFTWARE SYSTEM

The RGC Software System consists of routines programmed in a symbolic assembly language for execution on the simulation system described in Section 8.2. Except for format differences, these routines are identical to programs which might be implemented in an actual guidance and control computer to perform the executive, input/output, and reconfiguration functions.

Operation of an integrated multi-computer system presents some unique problems in the area of overall system control, sequencing, and mode/status maintenance. (The term, integrated system, is used here to exclude multi-computer systems which are essentially operated as a group of independent, non-interacting units.)

8.3 (Continued)

The major problems arise from considerations of the effect of processor failures on overall system performance and are primarily related to determining where to assign primary system control and how to reassign it in the event of a failure. The reassignment problem is particularly troublesome since reassignment implies a higher authority and the unit or function being reassigned is, by definition, itself the highest authority in the system. In systems where fault tolerance is not a primary concern, these problems are normally overcome by some variation of a "distributed" software executive. That is, the system level executive functions are shared by the multiple computers with some limited set of simple indicators (such as watchdog timer, parity checks, etc.) being used to trigger system degradation or reconfiguration.

In most systems designed primarily for fault-tolerance, a "hard-core" control unit is usually employed in one form or another which has responsibility for primary configuration control. Internal circuit redundancy is used to increase the survivability of this hard-core unit.

The philosophy of operation employed in the RGC software system is a logical extension of the concepts developed in the overall system design and stems from the desire to achieve extremely high failure detection/reconfiguration probabilities without the necessity for unique, special purpose, hard-core hardware. A redundant, majority-controlled software system is employed. Although it is resident in all computers, it is not"distributed" in the normal sense, since the entire function is duplicated identically and computed redundantly in all computers. Each computer, therefore, has equal status in terms of system control and decisions are achieved through a majority voting process. The highest level of system control - the decisions as to which computers are operational - is accomplished in the VCS's contained in each IOP. Each computer's opinion of the health of all computers is transmitted to the VCS's and the VCS logic performs an adaptive majority vote on this data. The result of this vote is, in turn, monitored by each computer and accepted as the current system status. Lower-level decisions are resolved through a software voting process which involves the computers exchanging opinions and accepting the resultant majority opinion.

During the Task 4 activity described in Section 4 of this volume, reduction of computational redundancy was investigated. This involves identifying levels of computational criticality related to the sensitivity of the external subsystem to loss of data from a particular computation function. Classifying functions in this manner would allow selective reduction of the number of computers required to redundantly compute given functions, from the three required for the most critical functions down to one for a non-critical function. However, the baseline software system developed during the study does not include this capability and all functions are assumed to be highly critical requiring a minimum of three parallel computations prior to system failures.

8.3.1 General

8.3.1.1 Theory of Operation. Critical computations, i.e., those computations falling within the "fail-operational" requirements, will be redundantly computed by up to three computers.* The redundantly computed data will be transmitted to a single VCS for voting/comparison and transmission to external subsystems. Up to three copies* of input data from external subsystems will be simultaneously received over multiple buses. The multiple copies will be received in all computers performing the computations. The VCSs and, hence, I/O buses in use at a given time, is determined by the status of those units and is not necessarily related to the computers currently assigned to the critical computations. The data input/output process is pictured in Figures 8-3 and 8-4.

8.3.1.1.1 Unit Status. Each unit in the system, computer, CPU, IOP, memory, or VCS, will have a status condition associated with it at all times. The three possible conditions are defined below:

1. Operational - A unit which has demonstrated no failures since last being brought on line.**

2. Conditionally Operational - A unit which has been implicated by voting discrepancies but which is able to pass all system tests.

3. Non-Operational - A unit which has suffered an identified permanent failure.

The minimum requirements for a computer to maintain operational status are:

1) Maintaining I/O sync with the critical computers as determined by the I/O TEST cycle.

2) Monitor and exchange of system status with all other operational computers.

3) True state of the corresponding P matrix diagonal element.

NOTE: Conditions 1 and 2 are necessary criteria for the other computers to vote for a true-state of the P matrix element corresponding to a given computer.

8.3.1.1.2 System Status. Each operational computer (where a computer is defined as the modules currently associated with a given IOP) has one of two system assignments at a given time.

Critical - An operational computer which is participating in the redundant critical computations. Three computers (if available) will be critical simultaneously.

*Reduction to less than three occurs only after two failures of computers or buses.
**A level of repeatability is required before a failure is identified in order to avoid erroneously reacting to transients.

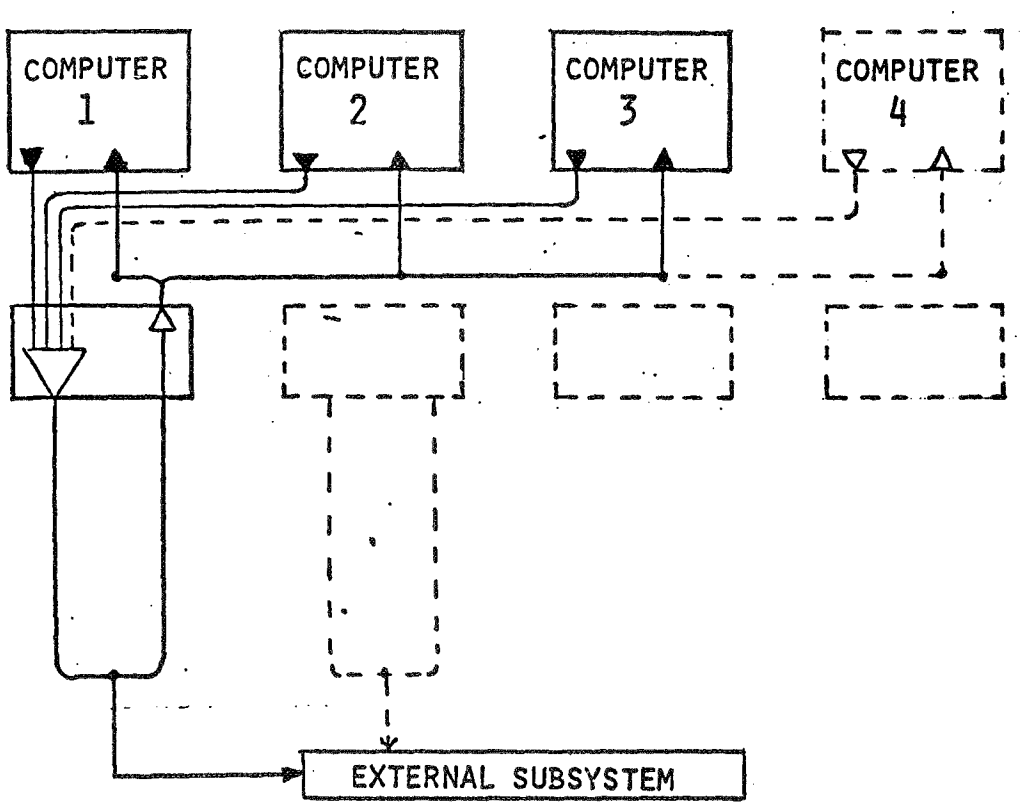


FIGURE 8-3. DATA OUTPUT

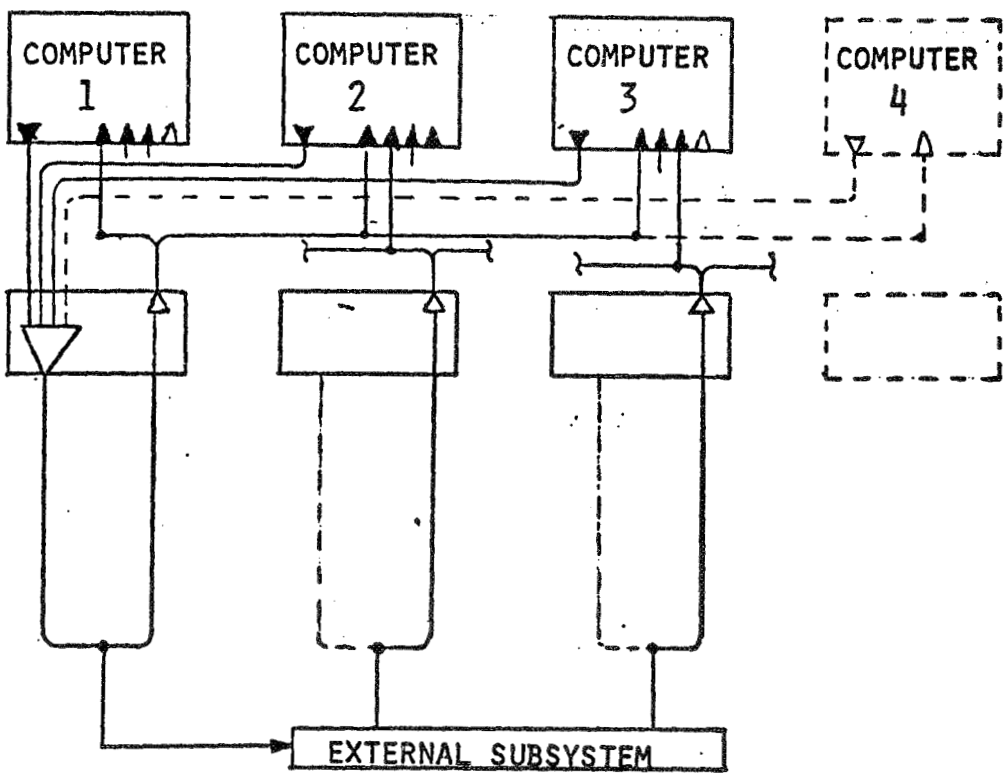


FIGURE 8-4. DATA INPUT

8.3.1.1.2 (Continued)

Spare - An operational computer which is maintaining I/O sync with the critical computers but which is not performing the critical computations. (There will only be a spare computer when all four computers are operational.)

Each operational VCS has one of four systems assignments at a given time.

Primary - The VCS (hence, system I/O bus) currently being used for output of all critical data.

Secondary - The "Backup" VCS which replaces the primary VCS in the event of a failure. This bus is used for the second copy of input data.

Associate - The bus being used for the third copy of input data.

Spare - An operational VCS which is not currently being used in critical I/O.

In Figures 8-3 and 8-4: Computers 1, 2, and 3 are Critical;
Computer 4 is Spare;
VCS 1 is Primary;
VCS 2 is Secondary;
VCS 3 is Associate; and,
VCS 4 is Spare.

8.3.1.1.3 System Timing and Synchronization. Each IOP has a separate real-time-clock (RTC). This clock is used as the real-time reference for the "computer" associated with each IOP. Synchronization of the four timing sources is accomplished by the Master Sync Controllers in the IOPs. Once the common timing reference is established, the IOPs maintain synchronization under IOP program control. This synchronization must be maintained within 16 μs for proper VCS operation. Out-of-sync conditions are detected by examining the voting status from the Primary and Secondary VCSes.

Synchronization between IOP and CPU is maintained on a rate (computational frequency) basis. That is, computations in the CPU at a given iteration rate are synchronized to the I/O cycle for that rate in the IOP. All computational rates are multiples of the RTC frequency. The Input/Output program is divided into four separate routines, RT1, RT2, RT4 and IOTEST; the first three perform the I/O for rates 1, 2, and 4 respectively, and the fourth performs a test function. One of these routines is executed each RTC interval. At the completion of each routine, a unique flag is set by the I/O program. When the CPU processes the RTC interrupt, it examines the four flags to determine which cycle has just been completed. This information is then used to initiate the next update of the appropriate computational frequency.

8.3.1.2 Software. The software system consists of three program modules: 1. Executive, 2. Input/Output program, and 3. Resource Con-

8.3.1.2 (Continued) troller. The Executive and Resource Controller are executed by the CPU; the Input/Output program is executed by the IOP. The following sections describe these programs in detail. They are described as if contained in a single computer, however it must be remembered that identical copies of these programs are executed simultaneously in all four computers.

Listings of these programs are included in Appendix 4 of this report.

8.3.2 Input/Output Program

The Input/Output program is executed by the IOP and transmits/receives data being supplied/processed by the CPU program. However, the Input/Output program is executed independent of the CPU and will sequence properly with or without the CPU data.

The I/O program is composed of four separate routines, RT1, RT2, RT4, and IOTEST. Receipt of the Real-Time Clock pulse (RTC) causes execution of one of these routines. Upon completion of the routine, the program normally enters the Idle mode to await the next RTC. However at the completion of IOTEST (every eighth RTC), the master sync process is initiated with all operational computers included in the sync process. In this case, receipt of the sync signal from the Master Sync Controller, initiates execution of the next program routine. The sequence of execution of the four routines is as follows: RT1, RT2, RT1, RT4, RT1, RT2, RT1, IOTEST, etc. It can be seen that the RT1, RT2, and RT4 routines are executed at 1/2, 1/4, and 1/8 of the RTC frequency, respectively. These rates correspond to the three fixed-interval computational frequencies available in the Executive Program in the CPU.

Each of the three routines performs the input/output functions required by the corresponding computational frequency. This consists of transmitting computed parameters to external subsystems, requesting input data from external subsystems, and transmitting "modifiable" data to the other computers. The term modifiable data is used to designate all the parameters necessary to completely define the state of a computational function(s). This data is necessary to initiate computations in a computer which is going through a transition from spare status to critical status.

One of four unique flags is set by the I/O program at the completion of each of the input/output routines. These flags are used by the Executive Program to synchronize the computational frequencies with the I/O data updates. Note that only the IOTEST routine is executed when a computer is the spare.

The IOTEST routine is executed every eighth cycle and is used to monitor the status of the VCSes and to perform test functions. The following status is sampled and saved for processing by the CPU.

1. S-matrix from Primary VCS - this sample determines whether any voting discrepancies have been detected over the last seven I/O cycles (RTC intervals).

8.3.2 (Continued)

2. P-matrix from Primary and Secondary VCSes - these samples determine the majority opinion as to the unit status of the four computers. Samples from two VCSes are taken to detect VCS failures.

3. R-matrix from Primary and Secondary VCSes - these samples are used to verify the VCS mode selection. Samples from two VCSes are taken to detect VCS failures.

Two test functions are performed; the VCS Test, and System Status Test.

VCS Test - This test is conducted by the critical computers. It consists of each computer transmitting one of two test messages to the Primary and Secondary VCSes. The messages are sent in a predetermined sequence of four combinations as follows:

1. Computer 1 - Message 1
   Computer 2 - Message 2
   Computer 3 - Message 2
   Computer 4 - Message 2

2. Computer 1 - Message 2
   Computer 2 - Message 1
   Computer 3 - Message 2
   Computer 4 - Message 2

3. Computer 1 - Message 2
   Computer 2 - Message 2
   Computer 3 - Message 1
   Computer 4 - Message 2

4. Computer 1 - Message 2
   Computer 2 - Message 2
   Computer 3 - Message 2
   Computer 4 - Message 1

One combination is transmitted during each test cycle in the I/O program; hence the test is completed every 32 cycles.

During the test, the VCSes retain the same voting mode as is currently being used for critical I/O. After the test messages are transmitted, the voting status (S-matrix) is sampled and saved for processing by the CPU. As

8.3.2 (Continued) can be seen from the message combinations, the test is used to verify proper detection and reporting of data discrepancies by the voting logic in the VCSes.

System Status Test - This test is conducted by all operational computers. It consists of each computer transmitting a system status message to the Primary VCS. The VCS is put into a mode to include all operational computers, and the message is addressed to a "dummy" LP address so that the voting function is performed. The system status message consists of the following data:

Operational Computers

Critical Computers

Operational VCSes

Primary VCS

Secondary VCS

Associate VCS

VCS Test Sequence Counter

The purpose of this test is twofold; 1) to provide a means of verifying I/O sync in a spare or transitional computer, and 2) to provide majority verification of the primary system status computed and maintained in each of the operational computers. After the message is transmitted, the voting status (S-matrix) is sampled and saved for processing by the CPU.

8.3.3 Executive Program

The Executive program is executed in the CPU and is responsible for task scheduling, synchronization of I/O data, maintenance of real-time reference, and time-critical status monitoring. All tasks to be executed by the CPU are divided into four groups depending on their frequency of execution; i.e., computation rate. The four groups are Rate 1, Rate 2, Rate 4, and Background. The first three groups represent precise iteration requirements; Rate 1 is executed at 1/2 the frequency of the RTC, Rate 2 at 1/4, and Rate 4 at 1/8. The Background group is executed whenever time is available and hence its iteration rate is variable. As previously indicated, four program flags set by the Input/Output program are used to initiate computations at each of the rates; hence, once completed, the tasks in Rate 1 will not be executed again until the RTC interrupt occurs and RT1 flag is set.

Within each rate, execution of the individual tasks occurs in a fixed, predetermined sequence. However, determination of which rate to execute is based on a priority structure where Rate 1 has the highest priority and Background has the lowest. Hence, once initiated all Rate 1 tasks will be completed without interruption but Rate 4 tasks may be interrupted by either Rate 2 or Rate 1. Obviously the total computational load must be such that each rate is completed prior to the next setting of its I/O flag.

8.3.3 (Continued)

At any given time, the parameter, RATE, contains the current rate being executed and each rate has status flags which determine the mode of that rate. The three possible modes are:

1) Done - all tasks in the rate have been completed. Nothing further is to be done until the appropriate I/O flag is set.

2) Interrupted - the rate was interrupted during execution by a higher priority rate.

3) Initialized - the I/O flag for the rate has been set but execution of tasks in the rate has not begun.

In the current system only three tasks are included in the schedule. These are Rate 1 Sample Task, Rate 2 Sample Task, and the Resource Controller. The tasks are executed at the Rate 1, Rate 2, and Rate 4 frequencies respectively, and the first two tasks are simple arithmetic and logical computations used to represent real-time computational tasks. The Resource Controller is described in the next section.

Every eighth I/O cycle is a test cycle indicated by the setting of the IOTEST flag. When this occurs the Executive executes a status update subroutine. This subroutine has two functions:

1. Examine the IOP flags which indicate the I/O transmission status. One of these flags is set if three successive faulty message transmissions are detected by the IOP. The states of the IOP flags are used to update four of the Resource Controller Error Status flags, E1VAS1, E2VAS1, E1VAS2, and E2VAS2.

2. Test for completion of a transition from non-critical status to critical status. If a transition has been completed, the system status data is updated to reflect the new status.

8.3.4 Resource Controller Program

The Resource Controller Program is composed of two major routines, the I/O Monitor and Test routine, IOMAT, and the System Configurator, RECONF.

8.3.4.1 Input/Output Monitor and Test Routine. This routine examines the status samples stored by the Input/Output program and uses this information to update the Resource Controller Error Status flags. The routine is executed at the Rate 4 frequency, the same frequency as IOTEST. The Error Status flags are described in Table 8-1.

8.3.4.2 System Configurator. This routine is executed at the Rate 4 frequency and is responsible for computing and maintaining the system status data. Once initialized, the system status will remain constant as long as all of the Error Status flags remain reset. When one or more of these flags is set, the Status Change Analysis subroutine, SCANAL, is executed. This subroutine uses a generalized pattern comparison structure to evaluate the change in status. Error patterns, i.e. particular combinations of Error Status flag settings, are stored in two data tables. The table, EPAT1, contains the 1-set flags in the pattern and the table, EPAT0, contains the 0-set flags. Flags not appearing in either the 1-set or 0-set entries are "don't care" elements for the particular pattern.

When a status change is detected, the pattern of Error Status flag settings is compared against the patterns stored in the error pattern tables. If a match is found, a unique subroutine associated with the particular error pattern is executed. This subroutine updates the system status to reflect the effect of the detected fault. Three subroutines are used to compute new system status configurations.

1. CPASIN - This subroutine computes the system status (critical or non-critical) for the four computers using their current unit status (operational or non-operational).

2. VCASIN - This subroutine computes the system status (Primary, Secondary, Associate) for the four VCSes using their current unit status (operational or non-operational) and the previous status assignments.

3. SSREST - This subroutine performs a software majority vote on its own system status data and the data received from the other three computers. The data resulting from the voting process is then used as the new system status. Note that only computers which have operational status are included in the voting process.

Table 8-2 is a list of the error patterns currently in the program and indicates the resultant action when a pattern match is found. It should be recognized that this is a minimum set and represents the initial baseline. Further evaluation of failure modes and fault isolation techniques would undoubtedly greatly expand both the number of patterns and the number of Error Status flags.

Table 8-1. Resource Controller Error Status Flags

FLAG	ROUTINE WHERE SET	MEANING
1. E1CVP	IOMAT	Indicates a disagreement in the critical voting during the last I/O cycle
2. E3CVP	IOMAT	Indicates critical voting disagreements in three successive cycles
3. E2SST	IOMAT	Indicates two successive disagreements in the System Status Test
4. E1LMAJ	IOMAT	Indicates inability to reach a majority consensus during critical voting
5. E2VTP	IOMAT	Indicates a discrepancy detected during VCS Test on Primary VCS in two successive test cycles
6. E2VT1P	IOMAT	
7. E2VTS	IOMAT	Indicates a discrepancy detected during VCS Test on Secondary VCS in two successive test cycles
8. E2VT1S	IOMAT	
9. E2VAS1	Executive	Indicates an IOP transmission mode switch was detected in two successive I/O cycles
10. E2VAS2	Executive	
11. E2PCOM	IOMAT	Indicates disagreement in P-matrix samples between Primary and Secondary VCSes
12. E2RCOM	IOMAT	Indicates disagreement in R-matrix samples between Primary and Secondary VCSes
13. COMOPC	IOMAT	Indicates that P-matrix sample from Primary VCS disagrees with system status data
14. MJXFLG	IOMAT	Indicates disagreement in System Status Test

Table 8-2 Resource Controller Error Patterns

Error Status Flags *														Resultant Action	Routines Used
1	2	3	4	5	6	7	8	9	10	11	12	13	14		
													1	Update System Status to majority consensus	SSREST
												1		Update System Status to reflect new P-matrix setting	CPASIN
											1			Fail either Primary or Secondary VCS depending on which one disagreed with current system status data	VCASIN
										1					
	0	0		1		0								Fail Primary VCS	VCASIN
	0	0		0		1								Fail Secondary VCS	VCASIN
		0			1		0							Fail Primary VCS	VCASIN
		0			0		1							Fail Secondary VCS	VCASIN
	1		0	1	1	1	1							Fail computer indicated by critical voting sample of S-matrix	CPASIN
	1		0	1	0	1	0								
	1		0	0	1	0	1								
	1		0	0	0	0	0								
			0					1						Fail Primary VCS	VCASIN
			0						1						
		1		1		1								Fail computer indicated by VCS Test sample of S-matrix	CPASIN
		1		0		0									

*A "1" indicates that the flag must be 1-set; "0" indicates 0-set; blank means "don't care"

8.4 SIMULATION ACTIVITIES

8.4.1 General

The underlying purpose for the development and use of the Simulation system was two-fold:

1. To provide a tool which could be used for development and evaluation of computer/software system designs oriented toward fault-tolerant system operation.

2. Use the tool to develop and evaluate the system design proposed to satisfy the FOOS requirements.

Use of the Simulation system during the course of the study involved four primary activities.

1. Refining and solidifying the functional design characteristics of the selected system being evaluated during the study.

2. Debugging and evaluating operation of the Simulation system.

3. Debugging and refining the software for the selected system design.

4. Evaluating overall system performance using fault simulation.

The simulation activities, particularly number four, are somewhat open-ended in that system evaluation and design refinement efforts can be extended almost indefinitely particularly in the area of fault simulation. The primary goal during this study was to provide a reasonable level of confidence in the feasibility and functional performance of the proposed system in terms of satisfying the FOOS requirements. Although some of the desired goals were not achieved in the area of fault simulation, it is felt that the primary goals were accomplished and the desired confidence attained.

8.4.2 System Debugging

As previously indicated, all three elements involved in the simulation process (simulator, simulated system, and software for the simulated system) were designed and developed in parallel during the study. Therefore a considerable amount of "interactive" debugging and design refinement were required in order to reach a point where total system operation was possible.

The Simulation system was actually developed in two stages due to parallel development of the simulator and simulated system required by the desired schedule. During the first stage, a general simulator system was developed which allowed for a large variation in the detailed design of the actual system to be simulated. The second stage involved implementing the detailed RGC computer system design in the simulator once that design

8.4.2 (Continued) had been reasonably well formalized. This approach worked reasonably well with one exception. The precision required in the simulation due to the IOP/VCS design, in terms of time resolution between simulated processors, was not anticipated. The timing structure implemented in the "general," first-stage version of the Simulator would not adequately represent the synchronous operation of the asynchronous IOPs (proper performance of the VCS depends on synchronous arrival of data from all IOPs, but the IOPs have independent clocks and operate on independent stored programs). In order to overcome the difficulty, the timing ring structure described previously was implemented.

The system design to be simulated was developed in considerable detail before it was implemented in the second-stage of the simulator development. In addition, the design is such that the unique, critical features are mainly contained in the IOP/VCS area and not scattered through the various elements of the system. Consequently, design modifications implemented during the debugging process were largely limited to the IOP and were primarily refinements in the VCS operation, such as the logic associated with transferring data from the IOP input channels to the VCS voting logic.

The philosophy of operation and general structure of the software for the simulated computer system were developed as an integral part of the overall system design; i.e., software considerations were applied during development of the system design not after the fact. For this reason, the concepts of system control, specifically the reliance on majority control of system configuration, are consistently applied throughout the design of both the hardware and software system. The primary problems discovered in debugging the software system were the normal varieties of programming "bugs."

8.4.3 System Operation

Operation of the overall system was evaluated in three phases:

1. Detailed operation of the IOP/VCS/bus system.
2. Fault-free operation of the total combined hardware/software system.
3. System operation with injection of simulated malfunctions.

The first phase consisted of simulating combinations of up to four IOPs. The individual VCS modes (4-way voter, 3-way voter, 2-way comparator, and selector) were selected and various combinations of data messages were simulated to verify the proper responses to both correct and incorrect data comparisons in each of the modes. In addition, the timing relationships between the data messages from the IOPs were varied to simulate marginal or out-of-sync conditions caused by timing variations or IOP failures. Examples of eight cases run in the 3-way voter mode are provided in Appendix 4.

8.4.3 (Continued)

The second phase of system evaluation involved simulating total system operation in the four possible system configurations reflecting the number of computers which are "operational." This phase was intended to verify operation of the hardware/software system in its steady-state modes of operation and to provide a baseline operation from which fault simulation could proceed. No significant problems were uncovered during this phase and it resulted primarily in further debugging of the software. An example of this phase is presented in Appendix 4.

The third phase involved injection of simulated faults in the system to verify proper detection and recovery. It should be realized that while this is the first point at which faults were simulated in an automatic, preplanned, manner, it was not the first time at which system operation was observed in the presence of faults. Both of the previous phases of evaluation represent a degree of fault simulation. In the first phase, faults were simulated by "pre-setting" erroneous data in the IOP messages and by using the "DLY" (delay) feature in the simulator to simulate IOP timing anomalies. During the second phase, the results of "unplanned" faults were observed in the process of debugging the software system and, in fact, the effect of apparent system failures caused by program "bugs" is a very important consideration in its own right, although not directly germane to FOOS considerations.

It was necessary to cut the fault simulation phase somewhat short of hoped-for goals due to funding considerations, however, a limited set of faults were simulated and successfully detected. The primary area where additional fault simulation was desired is in the IOP/VCS, since this area is critical to the design approach.

9.0 LOCAL PROCESSOR TRADE-OFFS AND DESIGN

9.1 INTRODUCTION

The purpose of this section is to report the results of evaluating various local processor options and their applicability to the Space Station Guidance and Control System. The need for local processing has been established from the overall system analysis and trade-offs. A common I/O data bus has been assumed to provide the means of exchanging data between the local processors and the central G&C computer complex. The purpose of the local processor is to perform computations associated with its subsystems, including on-board checkout and in-flight performance monitoring, and to provide a standardized digital interface with the data bus.

The local processor can be considered to consist of three (3) distinct sections as shown in Figure 9-1.

1. Interface with the data bus - called Standard Interface Unit (SIU)
2. Arithmetic processor and memory section for instruction and data storage - called Preprocessor.
3. I/O section providing interface with the subsystem electronics - called Electronic Interface Unit (EIU).

A general description of a candidate local processor design has been furnished by NASA for the purpose of evaluating its capability to perform the computational tasks. Tables 9-1 and 9-2 show the functional characteristics of this processor. It should be noted that the evaluation was constrained by the lack of more detailed descriptions of the candidate to an examination of the speed, word length, memory capacity and input/output characteristics.

In addition, the objective of this task is to consider alternate local processor design approaches to determine their merits. Modular designs and special purpose processors fall into this category. The final result of this study task is a functional description of the local processor design recommended for the application.

9.2 LOCAL PROCESSOR TRADE-OFFS

Trade-offs considered in the local processor design can be categorized as follows: (a) preprocessor trade-offs, and (b) EIU trade-offs. The trade-offs leading into SIU design have been conducted as part of the data bus design study.

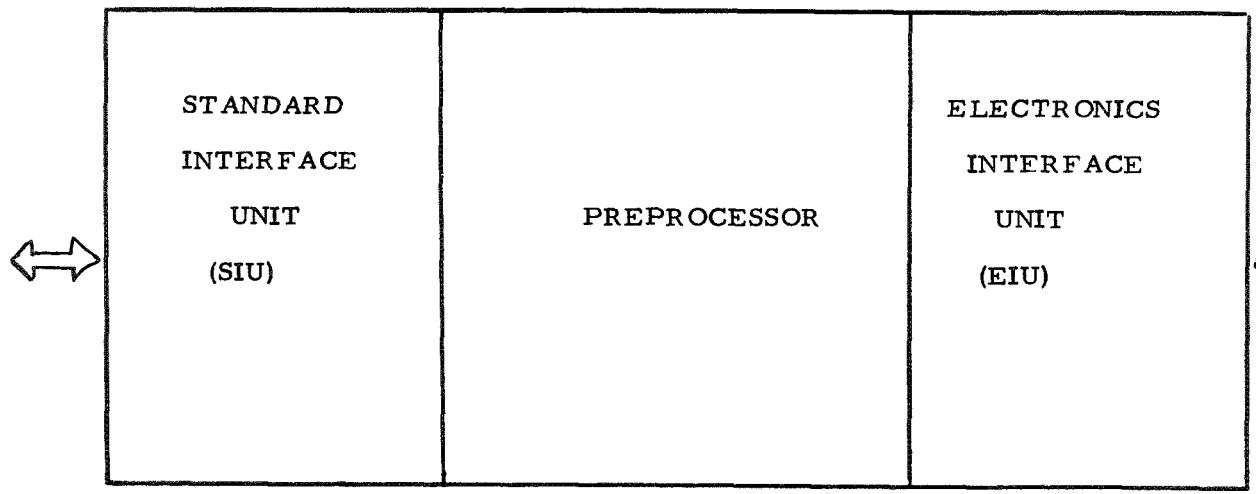


Figure 9-1. Local Processor Definition

TABLE 9-1

FUNCTIONAL CHARACTERISTICS OF THE CANDIDATE PREPROCESSOR

Word Length	16 bits
Memory Capacity	
Fixed (ROM)	1024/4096 words *
Scratchpad (RWM)	64/512 words *
Speed	
Add Time	10 μsec.
Multiply Time	20 μsec.
Registers	
Accumulator	
Utility	
Memory Data	
Memory Address	
Instruction Set **	
28 Register Manipulation and Control	
12 Input/Output Control	
Electronic Interface Unit	
1 16-bit Digital Parallel Channel (Input)	
1 16-bit Digital Parallel Channel (Output)	
12 Analog Channels (Input)	

* Original Specification/Modified Specification
** See Table 9-2 for basic instruction list

TABLE 9-2

BASIC INSTRUCTION LIST OF THE CANDIDATE PREPROCESSOR

Arithmetic Functions	Logic Functions	Control Functions
Add	And	Clear
Subtract	Or	Increment
Multiply	Exclusive or	Decrement
Divide	Complement	Condition
Absolute Value	Shift Right	Read
Double Precision Add and Subtract	Shift Left	Write
	Rotate	

9.2.1 Preprocessor Trade-offs

The preprocessor trade-offs are primarily concerned with the processor type (general purpose vs. special purpose), operating speed, word length and memory characteristics.

9.2.1.1 General Purpose vs. Special Purpose Preprocessors - Although the local processor requirements have been treated in terms of general purpose processing, special purpose processors should not be completely ignored. Since each special purpose processor can be designed for the specific application, total system weight, volume and power can be minimized. Also, the overall organization, instruction set and input/output section can be designed to fit the computational task, and thereby minimize and simplify the software.

Special purpose processors also have drawbacks. Non-recurring hardware costs will be higher because there will be many processors to be developed. Logistics and spare parts problems for the spacecraft would be greatly increased. Integration problems can also be very complex unless the interface standards such as data formats and power supplies are clearly defined at the beginning of the program and strictly enforced thereafter.

Two special purpose processors using incremental processing methods have been considered briefly in this study. These are a digital differential analyzer (DDA) and a Coordinate Rotation Digital Computer (CORDIC).

9.2.1.1 (continued)

DDA's have found numerous applications in guidance and navigation systems where the computational task consists primarily of a solution of differential equations and extrapolation equations. The general form of equations for the local processors is such that an exclusive DDA approach is highly inefficient and impractical. The potential advantage of a DDA is in a hybrid general purpose (GP)/DDA computer, where the DDA can be exploited for solution of continuous functions and the GP can be used for initializing and updating the DDA, decision making, mode switching, solution of complex but slowly varying functions and solution of non-continuous functions. By employing the hybrid approach, it is generally possible to substantially reduce the processing speed requirements of the GP processor below that necessary in an all GP computer. The major trade-off, therefore, lies between the relative complexities of the faster GP processor, and the slower hybrid GP/DDA.

The CORDIC processor similarly reduces the speed requirements of a GP processor if used in parallel with the GP to compute sine and cosine functions and to perform coordinate axes rotations. Again, this trade-off is between a fast GP processor and a slower GP processor/CORDIC hybrid system. Therefore, the GP vs. special purpose processor trade-off will be considered in conjunction with speed trade-offs in the following section.

TABLE 9-3

LOCAL PROCESSOR REQUIREMENTS

	Memory Requirements		
Subsystem	Read Only (words)	Read/Write (words)	Speed Requirements (Ops/Sec.)
SIRU	2520	330	382,500
OAS	2200	300	500
CMG	2950	570	61,400
RCS	2100	450	72,000

9.2.1.2 Speed Trade-Offs - An investigation of the candidate system defined in Table 9-3 indicates that the speed requirements vary over a wide spectrum. The highest speed requirement is imposed by the Strapdown Inertial Reference Unit - 382,500 short operations per second. One solution to this requirement is to use a single preprocessor design which is sufficiently fast for SIRU computations. The speed is within the state-of-the-art of aerospace computers using semiconductor memories, the cost and complexity of a fast computer (400,000 operations/sec.) is nearly the same as it is for a computer with one-fourth of the speed.

A second solution is to design one preprocessor to satisfy RCS, CMG and OAS speed requirements and to treat the SIRU separately. For the SIRU, one could achieve the speed by either (1) using multiples of the same processor, (2) provide a special purpose processor like a Digital Differential Analyzer (DDA) or a CORDIC processor to complement the basic preprocessor, or (3) transfer some computational load to the central G&C computer complex. However, all three alternatives have very serious disadvantages. The first one, splitting the functions between several parallel computers, requires three or four slow processors operating in parallel. The task of splitting computations between parallel processors appears feasible but introduces additional complexity and overhead in the software design. The prime disadvantage is the additional hardware required - considering that either triple or quadruple redundancy will be required, the total number of SIRU preprocessors could be as high as 16! This certainly is not the right approach to a system where high reliability is of utmost importance.

The second alternative appears to provide some gain in effective speed with either a hybrid GP/DDA or GP/CORDIC approach. Both systems could be mechanized with state-of-the-art MOS LSI circuits. A standard DDA integration circuit in a single chip has been developed by Autonetics. The device is a complete self-contained integrator/servo designed for use in parallel operation. Approximately 40 such integrators are required to mechanize a minimal processor for inertial platform control functions. For more sophisticated gimballed inertial navigation requiring an excessive number of integrators the CORDIC processor could be implemented with readily available shift registers and read-only memories.

However, it appears that a speed increase by a factor of four cannot be achieved with either approach. Considering the added hardware penalty and the disadvantages of special purpose design, no further investigations into GP/special purpose hybrid systems is considered necessary.

9.2.1.2 (continued) The third alternative does not provide any relief at the preprocessor level unless the computational load at the SIRU is reduced down to almost minimal level: instrument output filtering, failure detection and coordinate transformation. From the overall system consideration, this was not considered a desirable solution. Therefore, the first alternative, using a common preprocessor design of adequate speed to handle the most demanding computations, approximately 400,000 operations per second, appears to be the best choice.

9.2.1.3 Word Length Trade-Offs - Another key design parameter for the preprocessor is the word length. A 16 bit instruction word provides an adequate number of bits for defining instructions, specifying address modifiers and address field. For the data word, 16 bits has been judged adequate for all RCS and CMG computations. However, certain computations in position and attitude determination do require more than 16 bit precision. An investigation of the SIRU and OAS computation requirements was made to determine the extent and the frequency of such high precision computations and whether longer word length or double precision arithmetic capability is required.

Of prime concern was the need for double precision multiply capability because of its added complexity to the processor design. Simple double precision computations, such as add, subtract, store and fetch are relatively easy to implement and do not add much to the hardware complexity of the processor. Therefore, the analysis of SIRU computations was made assuming that the simple set of double precision instructions was included in the instruction set.

The results of the investigation show that practically all SIRU computations can be performed such that double precision multiply is not required. The one computation that would appear to need double precision multiplication is the direction cosine orthogonalization when multiplication of the elements of the C matrix occurs. This can be accomplished with a software double precision multiply routine. An analysis of the direction cosine update equations indicates that a double precision multiply is not required. An approximate equation for round-off error is

$$\sigma_r = (2.1 \times 10^5)\, 2^{-n} \sqrt{Tf} \qquad (9.1)$$

where
- n is the word length in bits
- T is the time the round-off error has to accumulate in seconds
- f is the frequency of computation in times/second
- σ_r is the attitude error in arc seconds due to the computations

9.2.1.3 (continued)

and 2.1×10^5 is the number of arc seconds per radian

if $r = 4$, $T = 1000$, $f = 100$, then $n = 24$ so that a word length of 24 bits or more should be used to represent the direction cosine matrix under these conditions. The updating equation for a second order algorithm has the form

$$C_{n+1} = (I + \Delta\theta + \Delta\theta^2/2)\, Cn \qquad (9.2)$$

where I is an identity matrix, and $\Delta\theta$ is a matrix of accumulated pulse counts as follows:

$$\Delta\theta = \begin{bmatrix} 0 & \Delta\theta z & -\Delta\theta y \\ \Delta\theta z & 0 & \Delta\theta x \\ \Delta\theta y & -\Delta\theta x & 0 \end{bmatrix} \qquad (9.3)$$

If the maximum pulse rate is 10,000 pulses/second and if the pulse accumulation registers are emptied every .01 seconds, the maximum value of $\Delta\theta x$, $\Delta\theta y$, or $\Delta\theta z$ is 100. This quantity can be represented by 8 bits including a sign bit.

Computation of equation 9.2 by the sequence

$$A = I + \Delta\theta + \Delta\theta^2/2 \qquad (9.4)$$

$$C_{n+1} = AC_n \qquad (9.5)$$

would require double precision multiplication to preserve 24 bits of accuracy. A computation sequence that preserves 24 bits of accuracy and only uses 16 bit multiplication is:

$$A = \Delta\theta C_n/2 \qquad (9.6)$$

$$B = \Delta\theta(C_n + A) \qquad (9.7)$$

$$C_{n+1} = C_n + B \qquad (9.8)$$

This is possible since the pulse count for θ can be represented by only 8 bits of a 16 bit word. The quantity A, using a 16 bit representation for C_n in equation 9.6, can be shifted such that its meaningful data, after the 16 bit $\Delta\theta C_n$ multiply, is in bit locations 9-24. The $C_n + A$ addition of 9.7 can be performed in double precision and the shifting, as for A, used for B in equation 9.7 to give significant data in bit locations 9-24 of B. A double precision addition of B to C_n in equation 9.8 gives a C_{n+1} accurate to 24 bits.

9.2.1.3 (continued)

Another computation that needs double precision operations is the filtering of the instrument outputs and then only for the long term filter and under the unlikely condition that the 10,000 pulses per second are all unidirectional and must be summed for 60 seconds or longer. This requires representation of numbers on the order of 600,000 which requires a 21 bit word (including sign).

Considering the Optical Attitude Sensor, the only computations requiring more than 16 bits are those for the computation of star tracker pointing angles and these are dependent on the attitude accuracy requirements. If the required attitude accuracy is $.01^{\circ}$ and the error due to star tracker computations are to be one-tenth of this, then numbers on the order of 1 part in 360,000 need to be represented which would require 19 bits.

Horizon sensor measurements can be made to an accuracy of about $.1^{\circ}$ and if computational errors are to be one-tenth of this quantity, numbers on the order of 1 part in 9000 need to be represented which only requires 14 bits.

The foregoing analysis indicates that a 16-bit data word is adequate for the subsystems investigated. Double precision multiplication requirements can be minimized by careful scaling of the quantities and proper shifting of data to preserve significant bits. Limited double precision capability for adding, subtracting, storing and fetching is required in the instruction set.

9.2.1.4 Memory Trade-Offs - While the requirements of the central G&C computer complex indicate a definite need for some form of a magnetic main memory, the local processor memory can be best implemented with a combination of a read-only and read-write MOS semiconductor memories. The two prime objections to a semiconductor memory in the central G&C computer complex are: (1) volatility of the read-write memory and (2) inability to alter electrically the content of the read-only memory. These two arguments do not apply to the local processor for the following reasons.

Each local processor performs only functions dedicated to a specific subsystem which are defined relatively early in the development program and, once firmed up, change very seldom permitting the processor program to be committed to a read-only memory with a fixed interconnection pattern. An overlay mask is used for encoding the MOS ROM device with program information. The encoding mask is automatically generated by a special computer program, resulting in turn around times as short as two days, permitting changes in the program to be implemented with minimal delays. Once the encoding mask is generated, the device is fabricated in a standard production run.

9.2.1.4 (continued)

The volatility of the read-write scratchpad memory is not critical in the local processor because the central computer complex is capable of retaining the last set of computed data and can reinitiate the local processor program with minimum interruption in case of power transients.

MOS semiconductor memory offers several advantages in the local processor. Among the major advantages are lower size, weight, power and cost than magnetic memory, especially in small capacities required for the local processor. The MOS semiconductor memory will operate with the same clock as the logic circuitry and does not require additional interface and timing circuits. The cost of a semiconductor memory is linear with the capacity, which means that semiconductor memories can be broken into any size modules without cost penalty. Magnetic memories, on the other hand, require that a large magnetic array size be driven by a few electronic circuits to be economical.

Present state-of-the-art memory technology permits bit densities as high as 4096 for ROM's and 512 for RWM's. A typical memory of 3584 words ROM and 512 words of RWM can be mechanized with 14 ROM devices, 16 RWM devices and 2 address decoder circuits. Detailed characteristics of these devices are presented in Section 2.

9.2.2 Electronic Interface Unit (EIU) Trade-Offs

The studies concerned with the EIU have been based on the analysis of signal interface requirements. These interface requirements are presented in Table 9-4. Because of the lack of detailed description of the subsystem hardware, the interface between the local processor and subsystem electronics has been based on certain assumptions. These assumptions are summarized briefly in the following paragraphs.

TABLE 9-4

LP - TO - SUBSYSTEM INTERFACE REQUIREMENTS

SUBSYSTEM	OUTPUTS		INPUTS		
	Whole Word	Discrete	Whole Word	Discrete	Analog
SIRU		12 *	12		43 *
OAS	4	2	2	1	1 + 10 *
CMG	6 **		6		30 *
RCS		16			30 *
RCS (Central)		8			16 *

* For Test and Monitor Purposes Only
** Could be Analog

9.2.2.1 SIRU Interface - The instrument outputs are $\Delta\theta$ and ΔV in incremental form - a total of twelve incremental inputs. Since the SIRU is the only subsystem that provides incremental inputs to the local processor, it is considered more cost effective to locate the precounters for the instrument outputs in the SIRU electronics package rather than in the EIU of the local processor. The contents of these precounters will be periodically read into the local processor through a digital, whole-word input channel. The remaining inputs and outputs are used primarily for test and performance monitoring purposes.

9.2.2.2 OAS Interface - For this subsystem, the pointing angle commands and angular readouts are assumed to be in digital form. Again, the bulk of the analog signals are voltage and temperature monitoring signals.

9.2.2.3 RCS Interface - A valve control scheme using quad redundant valves shown in Figure 9-2 is assumed for the study. The scheme permits full operational capability after local processor failure in the engine station. All valve controls are discrete signals at bipolar logic levels, requiring the power amplifiers to be located at the valve controls. The analog signals monitor temperature, pressure and flow rates.

9.2.2.4 CMG Interface - The CMG interface consists primarily of six gimbal angle inputs and six gimbal rate outputs. The inputs are assumed to be digital, whole-word signals requiring a resolution of 12 bits. The outputs could be either in digital or analog form. Since no other subsystems have analog output requirements, it is recommended that digital interface be used for gimbal rate outputs.

9.2.2.5 Standardized EIU - An analysis of Table 9-4 indicates that the functional interface requirements for the four subsystems considered do not vary as widely as one may initially expect. If one also considers the data rates estimated for the local processor, the following conclusions can be readily drawn:

1. Data rate requirements are very low compared to the pre-processor I/O capability.

2. Because the local processor is dedicated to one subsystem, all I/O events can be initiated under local processor control. No need exists for the local processor to accept data from many sources asynchronously.

3. It is feasible to meet the different EIU requirements with one standardized EIU design. A modular standardized building block approach does not appear to offer any advantages over a single design because the requirements do not vary widely enough.

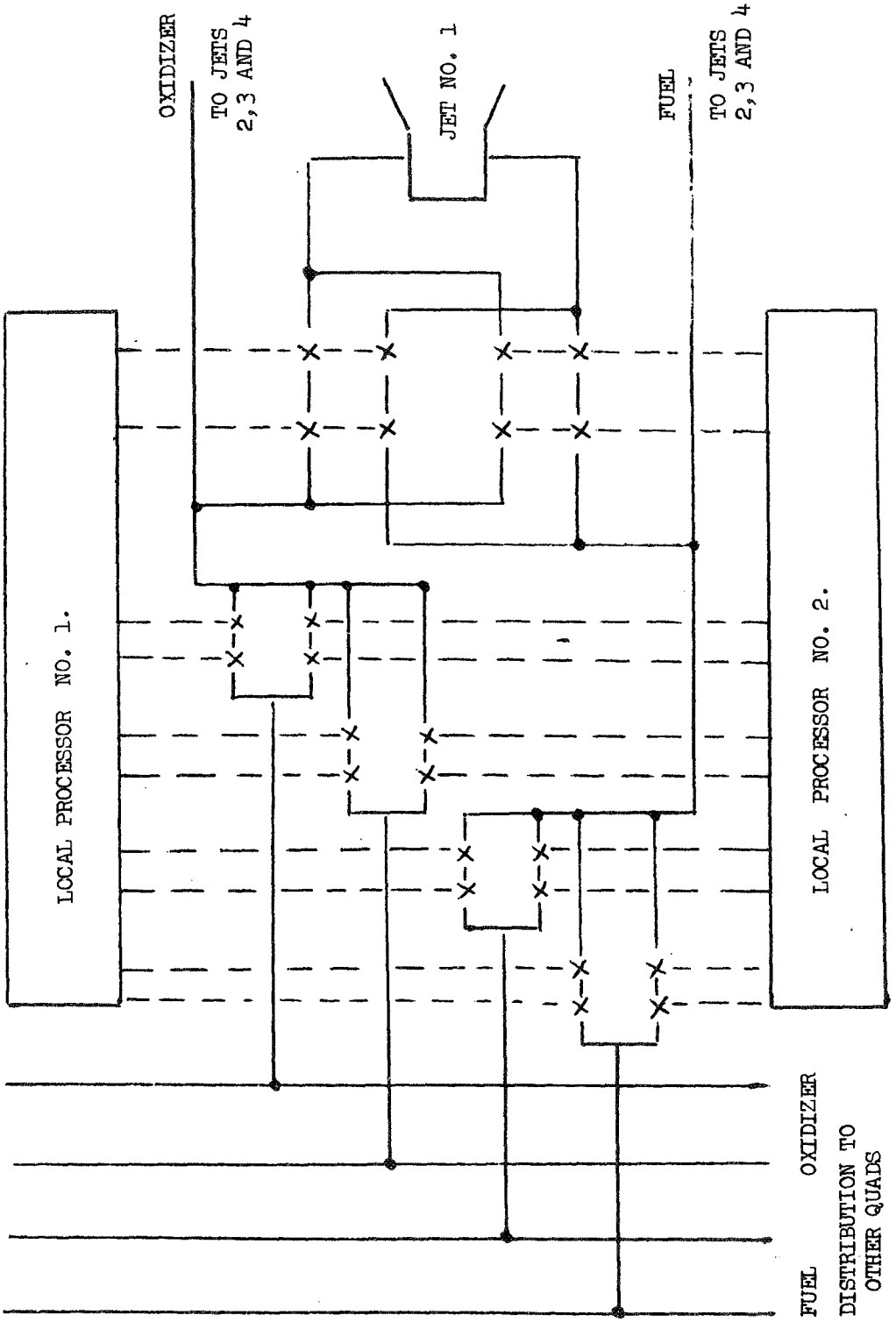


Figure 9-2. RCS Valving Controls

9.2.2.5 (continued)

4. The standardized EIU can be broken into two main sections: a digital section (input and output) and an analog section (input only). The following section describes the results of mechanization trade-offs for these two sections.

9.2.2.5.1 Digital EIU Section - Two basic considerations in the design of the digital section of the EIU are: (1) form of data transmission (serial or parallel), (2) method of implementing channel control.

Because of the low data rates, serial data transmission could be used for implementing digital data channels in the EIU. The basic advantage of serial transmissions is reduced number of line drivers, receivers and interconnections. The main disadvantage is the limited data rate capability. Although the overall data rates are quite low, there may be times when there exists a need to transmit several pieces of data at high rate. Therefore, use of serial channels may limit the usefulness of the local processor for other subsystems not investigated under this contract. Therefore, it is recommended that the data transfers take place in parallel in order to assure the usefulness of the local processor in a wide range of applications.

There are basically three methods used for controlling data transfers in aerospace computers: (a) programmed data transfer, (b) direct memory access (DMA), and (c) multiplexer channel.

The first method is the slowest but has the advantage of flexibility. Some form of interrupt is required for this method. Data transfer takes place between an I/O bus and one of the processor registers or a memory location specified on the I/O instruction.

Direct memory access provides data transfer between memory and external devices by "stealing" memory cycles from the processor program. This type of transfer is quite fast and its maximum data rate is limited to the memory speed. Word count and memory address registers are required at the EIU or subsystem electronics.

A multiplexer channel would provide the DMA with capability to sustain several I/O operations on a time-shared basis. The channel services the peripheral devices asynchronously as the input data becomes available or when the receiving devices can receive data. The use of a multiplexed I/O channel requires a buffer mode of data transfer. This mode uses stored control words and assigns areas in memory as input/output buffering areas. These areas are under control of the programmer.

9.2.2.5.1 (continued)

Considering that the local processor interfaces with only one subsystem and rates are quite low, the programmed data transfer method is quite adequate, requires minimum amount of hardware and offers most flexibility. It is the recommended method.

9.2.2.5.2 Analog EIU Section - There are two approaches to locating interface conversion equipment. The first distributes the conversion on operations to the sensors where the signals are generated. The various signals are converted to a standard digital format and transmitted to the digital section of the EIU. This means that each sensor must have its own analog-to-digital converter (ADC), a digital register and some logic to read out its contents upon computer command. This approach has several advantages and disadvantages. Among the advantages is that (a) single point failures can be isolated to the specific sensor, (b) signal grounds can be isolated resulting in reduced noise effects, and (c) the ADC can be operated at slow speed commensurate with the associated sensor data rate and (d) full advantage can be taken of future development of digital sensors. The major disadvantage of this approach is that it requires more hardware resulting in higher system complexity, weight and cost.

The second approach is to integrate the conversion equipment in the EIU section of the local processor. In this case the signals are transmitted in analog form to a multiplexer and encoded in the ADC located in the EIU. Time sharing of conversion equipment is possible with this approach; hence, hardware costs are reduced.

Considering the large number of analog signals and the relatively slow sampling rates, the second approach definitely results in reduced hardware cost and complexity. Therefore a time shared ADC in the EIU section of the local processor is recommended.

Another important factor in the EIU design is the determination of the location of the multiplexer switch. It is usually determined by the sensor distribution. If many sensors are located in one source area, the multiplexer is located in that area to save wires and line drivers and receivers. Otherwise, the multiplexer is located in or near the receiver, or EIU in this case. Sometimes a compromise is made and the multiplexer is broken into several sections that are located near sensor areas. Then the outputs of these multiplexers are multiplexed again at the ADC input. The location of the sensors in the Space Station can vary widely between subsystems. For example, in the SIRU all sensors are located in close priximity while in the RCS subsystem the temperature and pressure gauges are widely separated physically. It is important that the EIU have enough flexibility to interface efficiently with all subsystems and therefore a multiplexer at the EIU is recommended with additional discrete signals being provided for multiplexing control at the subsystem electronics.

9.3 CANDIDATE LOCAL PROCESSOR EVALUATION

An evaluation of the candidate preprocessor described in Tables 9-1 and 9-2 was made by comparing the functional characteristics against the requirements. The most critical limitation of the candidate preprocessor is the speed. An improvement by a factor of four is required in order to satisfy SIRU requirements. The problemsof using a slow preprocessor have been treated in Section 9.2.1.2 and will not be repeated in this section. In addition, the lack of discretes and/or interrupts may present some difficulties in interfacing with the subsystem electronics. Eight (8) analog channels are adequate if some multiplexing at the subsystem electronics is provided. A description of the recommended local processor design is presented in the next section.

9.4 RECOMMENDED LOCAL PROCESSOR DESCRIPTION

The LP is a programmable, parallel, digital machine utilizing a semiconductor memory, MOS logic, and having an input/output section that can be tailored to the application. The LP is to interface with the type 2 data bus that is described in the reports on the reconfigurable G&C computer. The functional areas of the LP are:

a) Central Processing Unit (CPU)
b) Memory
c) Electronic Interface Unit (EIU)
d) Power Converter
e) Clock
f) Standard Interface Unit (SIU)

Figure 9-3 is a block diagram of the LP. The LP interface with the subsystem will consist of DC voltage analog and discrete type signals. The discrete signals can be pulse or on/off types.

9.4.1 Central Processing Unit

The CPU operates on an internally stored program made up of 16-bit instructions. The basic logic speed is one megahertz. All operations in the CPU are done in parallel format as far as possible to attain maximum computing speed.

Data is handled in ordinary fixed point binary format with negative numbers expressed in two's complement. Data words are 16 bits long including sign. Double precision operations are possible in which the data word is 31 bits long including sign.

The CPU is organized into the following functional blocks:

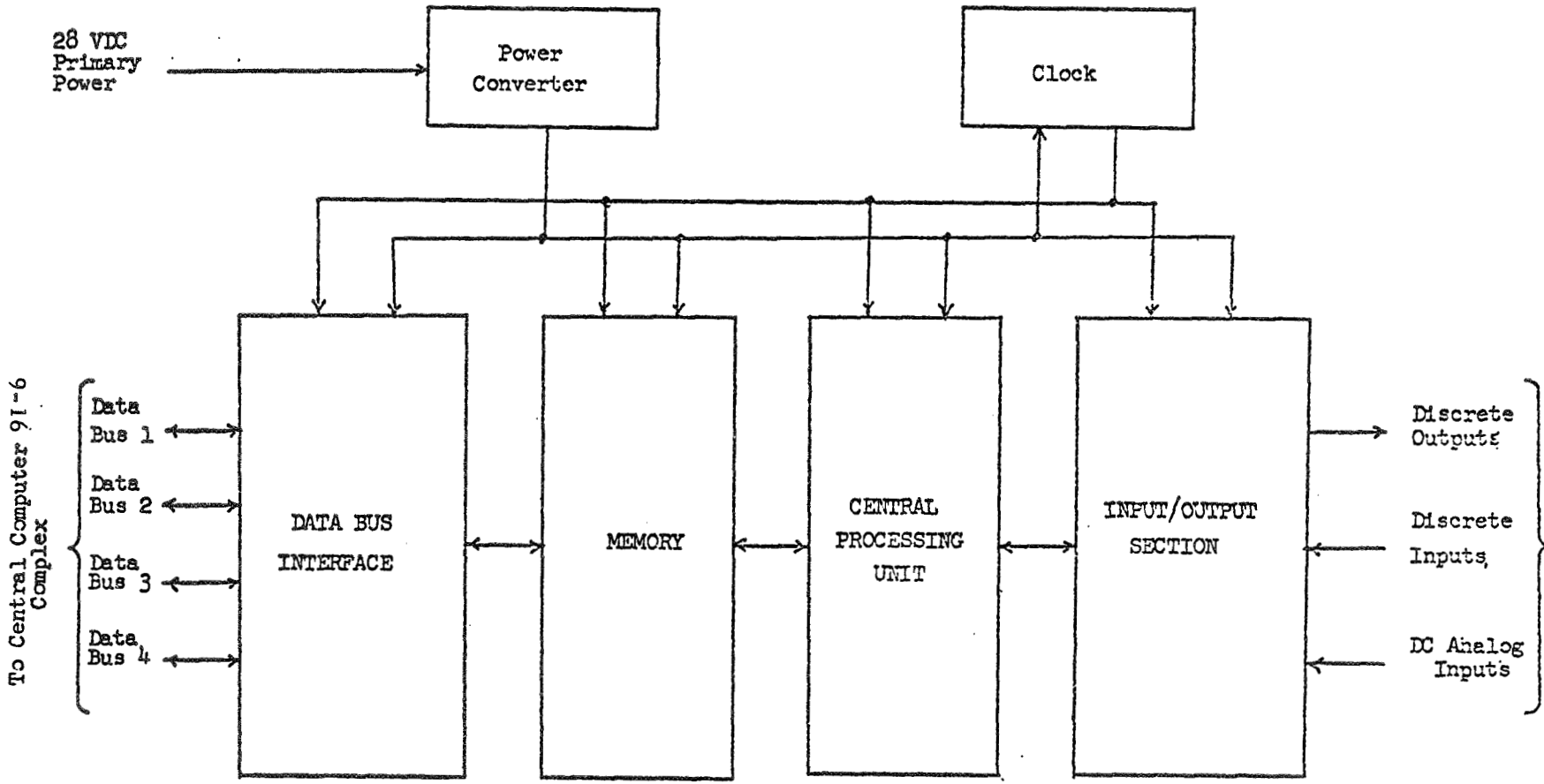


Figure 9-3. Local Processor Block Diagram

9.4.1 (continued)

a) Register File:	General register storage (each register is 16 bits long).
b) Program Counter (13):	Controls instruction access from memory.
c) Accumulator (16):	General purpose data register.
d) Extension (16):	General purpose data register.
e) Buffer (16):	Memory data buffer register.
f) Instruction Register (16):	Holds instruction being executed.
g) Control "A" (5):	Controls multiply and divide operations.
h) Shift Matrix (5):	Controls shifting operations.
i) Address Register (16):	Holds memory address of operands.
j) Operation Register (4):	Holds operation code.
k) File Address Register (9):	Controls general register operation.
l) Condition Register (4):	Holds results of comparison operations.
m) Auxiliary Register (16):	General purpose data register.
n) Adder (16):	Parallel adder/subtractor.

The number in parentheses indicates the size of the function in terms of the number of flip flops required to mechanize the function. A block diagram of the CPU is shown in Figure 9-4.

The CPU has a flexible instruction repertoire that includes the following:

a) Arithmetic (Operands in memory and/or registers)
- Add
- Subtract
- Multiply
- Divide

b) Comparison
- Compare registers
- Compare register and memory
- Branch on condition

c) Shift
- Shift left
- Shift right

d) Double Precision
- Add
- Subtract
- Fetch
- Store

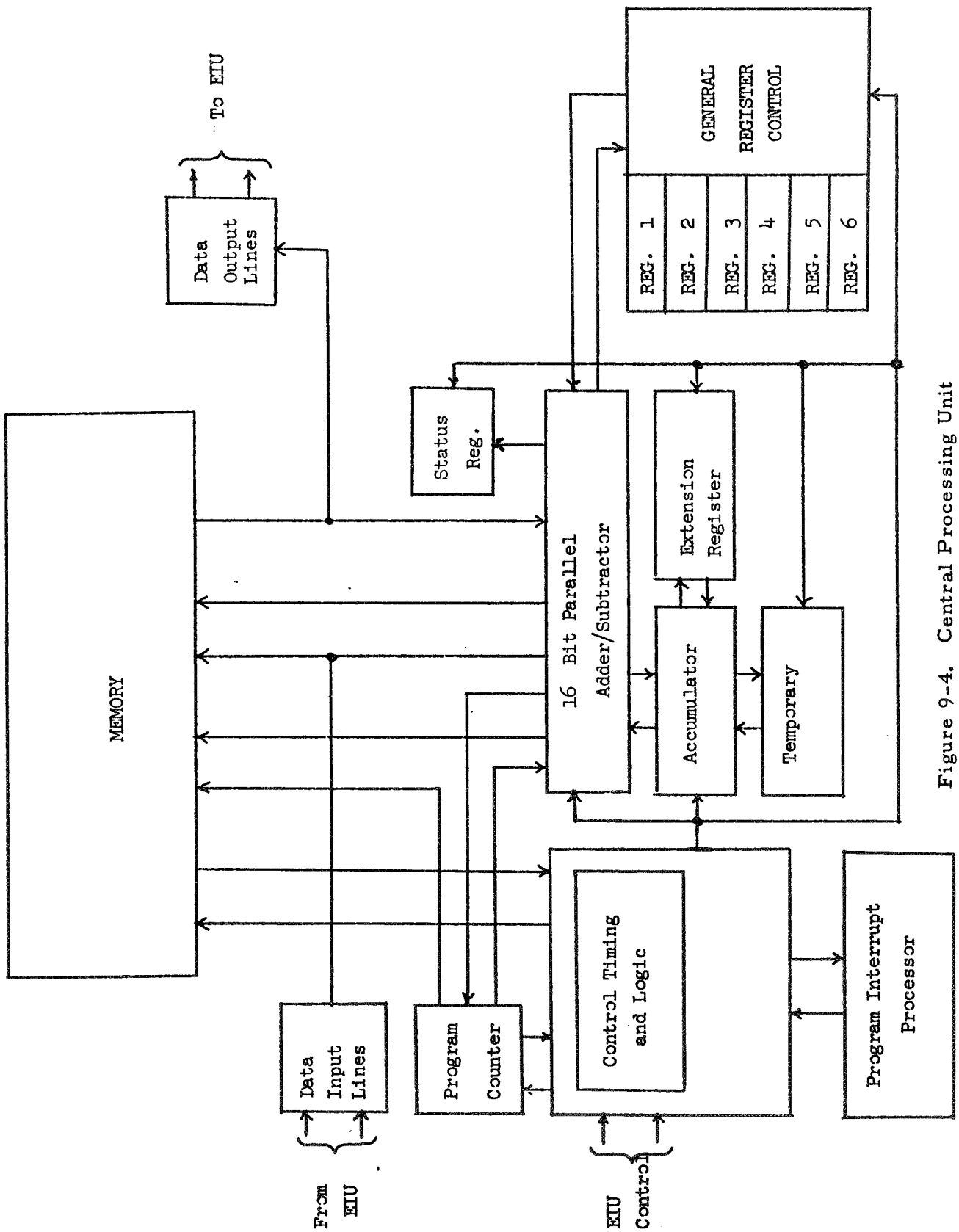


Figure 9-4. Central Processing Unit

9.4.1 (continued)

e) Data Movement
- Fetch memory
- Store in memory
- Exchange registers

f) I/O Control
- Set output discrete Group A
- Set output discrete Group B
- Read input discrete Group A
- Read input discrete Group B
- Input parallel data word
- Output parallel data word
- Disable interrupts
- Enable interrupts
- Read analog input X (Coding identifies one of 8)
- Read analog input group in sequence.

The CPU has two internal interrupts, one from the SIU and one from the EIU. The interrupt from the EIU indicates that the EIU has completed the commanded analog-to-digital conversion and the data is ready for the CPU. The CPU branches to a subroutine to take the data word from the EIU and place it in memory. The CPU then returns to the point of interruption in the program.

The interrupt from the SIU has two functions depending upon where the CPU is in the program. When the CPU is executing the main program, the interrupt indicates the reception of a control word by the SIU. The CPU branches to a dedicated location in memory to start an input or output routine and sets the interrupt false. All pertinent data is saved so that the CPU can return to the point in the main program where the CPU was interrupted. Next, the control word in the SIU is read into the CPU. The address field is placed in the program counter and that location accessed.

The content of that location is the first instruction of a subroutine that will handle the data transfer. A counter is formed by storing the number-of-words field into memory.

The CPU idles until the SIU interrupt goes true. The CPU sets the interrupt false and then accesses memory for a data word and places the word in a buffer in the SIU or reads the data word from the SIU buffer and stores the word in memory depending upon whether the operation is an input or output. In either case, after transferring the data word, the CPU increments the memory address register and decrements the number-of-words counter. If the counter is non-zero, the CPU idles until the SIU interrupt goes true to repeat the above operations. If the counter is zero, The CPU restores conditions prior to the interrupt and continues with the main program.

9.4.1 (continued)

In addition to the internal interrupts, the CPU has a minimum of four external interrupts. Each external interrupt will force the CPU to branch to a specified location for the next instruction. Each interrupt has a dedicated memory location assigned to it. A simple priority scheme that the lowest numbered interrupt has the higher priority provides orderly processing of the interrupts. The CPU automatically disables all other interrupts until it finishes processing an earlier or higher priority interrupt.

9.4.2 Memory

The memory is made up of a read only section and a read/write section. The read only section has memory addresses 0512 to 4095 and the read/write section has memory addresses 0000 through 0511. The memory uses MOS technology in its mechanization. Figure 9-5 shows a block diagram of the memory.

The memory contains 4096 words each 16 bits long. Each word is addressable on a random access basis. Words are written into or read from the memory in parallel. A memory read cycle time is a maximum of 750 nanoseconds and a memory write cycle time is a maximum of one microsecond.

9.4.3 Electronic Interface Unit

The EIU is made up of four areas:

a) Discrete output area
b) Discrete input area
c) Analog input area
d) Parallel data bus area

All operations of the EIU are initiated and controlled by execution of instructions by the CPU. Data transfer between the EIU and the rest of the LP is through the registers of the CPU.

9.4.3.1 Discrete Output Area - The discrete output area of the EIU consists of circuitry for a minimum of 32 discrete output signals divided into two identical groups. A buffer register having one bit position for each output holds the data received from the CPU. The outputs of this register are conditioned by line drivers and placed on the output lines. The output signals are complementary types requiring two wires per signal. The voltages on these lines are always complements of each other. True and false conditions for the signal represented by the voltage states of the lines can be arbitrarily defined. The voltage levels used are +5 VDC and ground. The line drivers are bipolar circuits to handle drive requirements. The rest of the circuits are MOS type circuits.

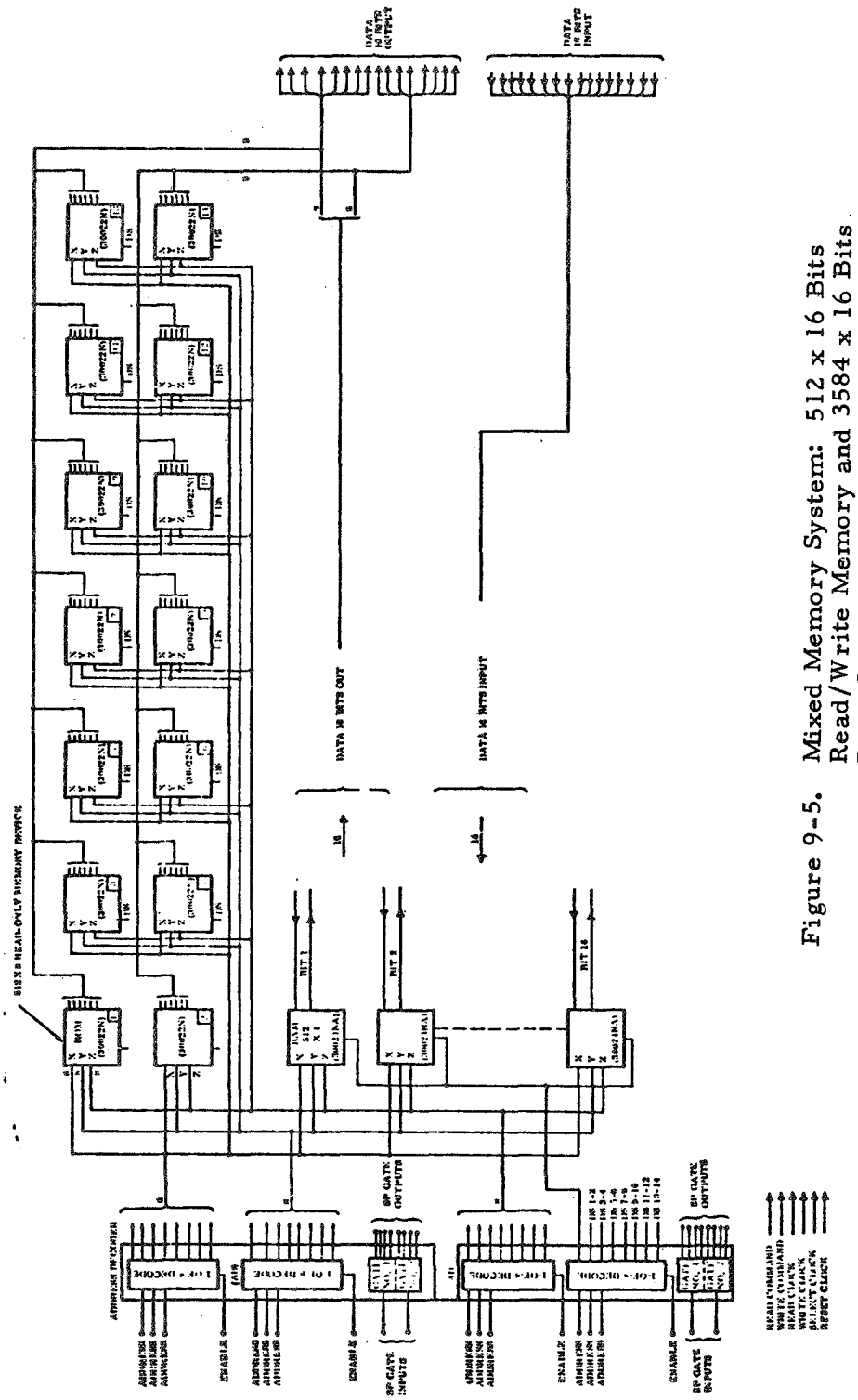


Figure 9-5. Mixed Memory System: 512 x 16 Bits Read/Write Memory and 3584 x 16 Bits Read Only Memory

9.4.3.1 (continued) Discrete outputs are organized in groups of 16 to conform to the LP internal word size. When used as a pulse type discrete, these outputs can be driven to give 250,000 pulses per second maximum. Single pulse outputs can have pulse widths as narrow as four microseconds. Figure 9-6 shows a block diagram of the discrete output area.

9.4.3.2 Discrete Input Area - The discrete input area is capable of receiving a minimum of 32 discrete type signals. These input signals are complementary types as discussed above. The signals on the input lines are conditioned by line receivers and strobed into a buffer register upon command from the CPU. All circuits are MOS type except for the line receivers which are bipolar. The input discretes are organized into groups of 16 to conform to the LP internal word size. Figure 9-7 shows a block diagram of the discrete input area.

9.4.3.3 Analog Input Area - The EIU has the capability of accepting up to eight d-c analog voltage inputs and converting these voltages into a twelve bit digital number including sign. The conversion time is four microseconds per bit plus five microseconds for settling time or a total of 53 microseconds per input.

An input filter is provided on each input line to prevent surges on these lines immediately after the input multiplexer switch is closed thus assuring accurate measurements. Overvoltage and short circuit protection is employed on the inputs to help prevent faults from propagating from one subsystem to another.

Figure 9-8 shows the basic block diagram of the converter. Initially, the register is set to zero, causing the output of the DAC ladder network to be equal to zero volts, then one of the series input switches is closed connecting an input line to the comparator amplifier. After sufficient time to allow for input switch closure, the bits of the register (most significant first) are sequentially set and reset depending upon the polarity of the output of the comparator amplifier. In this manner, the converter makes a sequential convergence on the analog input voltage until the number in the register corresponds to the input voltage to a resolution of $\pm 1/2$ of the least significant bit.

Reference voltages for the ladder network are supplied by a precision power supply to insure a high accuracy in the conversion procedure.

Conversion is initiated by the CPU executing one of the read analog input instructions. After completing the conversion, the digital number is held in a buffer register and the CPU notified by an interrupt. If a group of analog input conversions are desired, the CPU executes the read analog input group in sequence instruction. The converter starts the operation by converting analog input one. After setting the interrupt true,

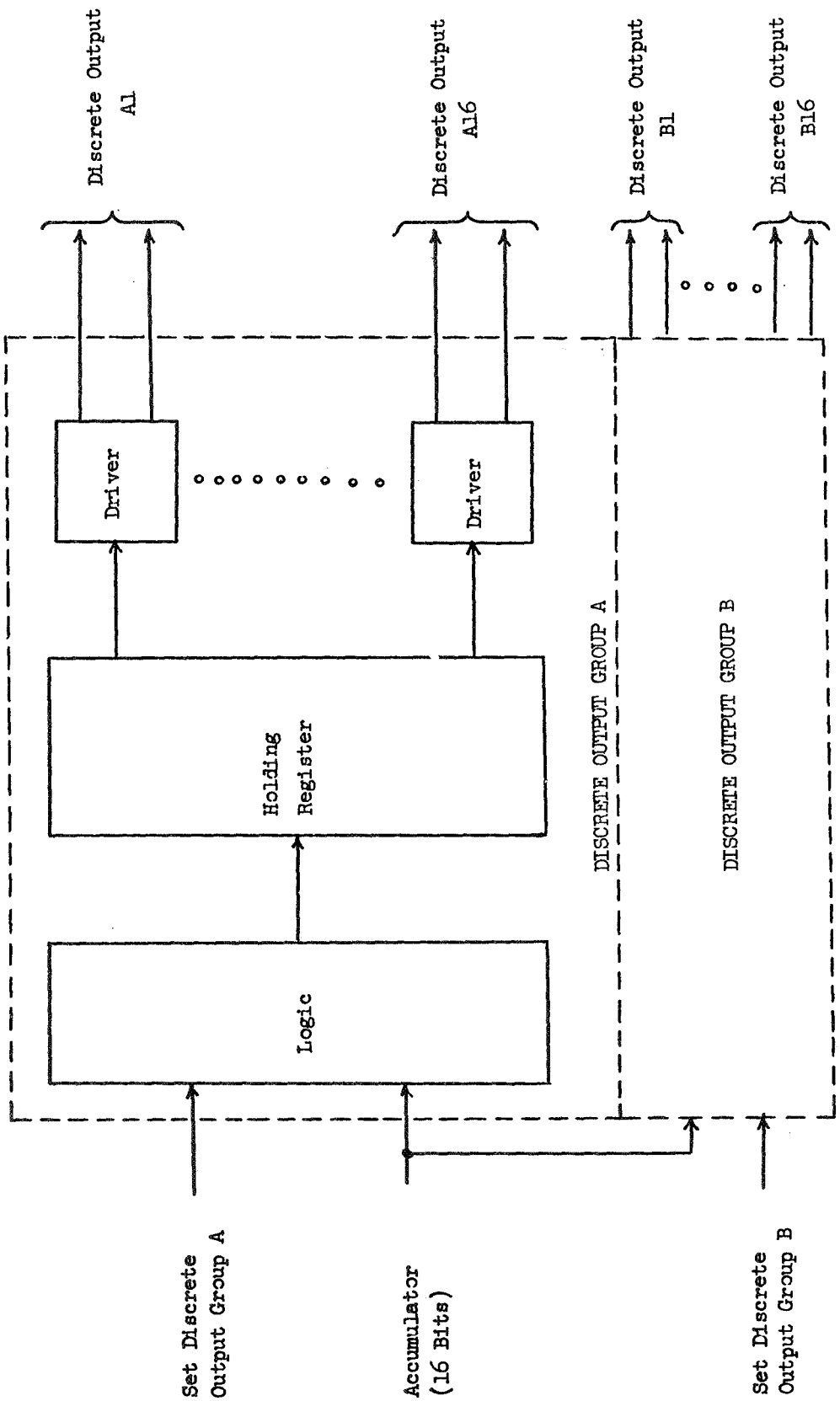


Figure 9-6. Discrete Outputs

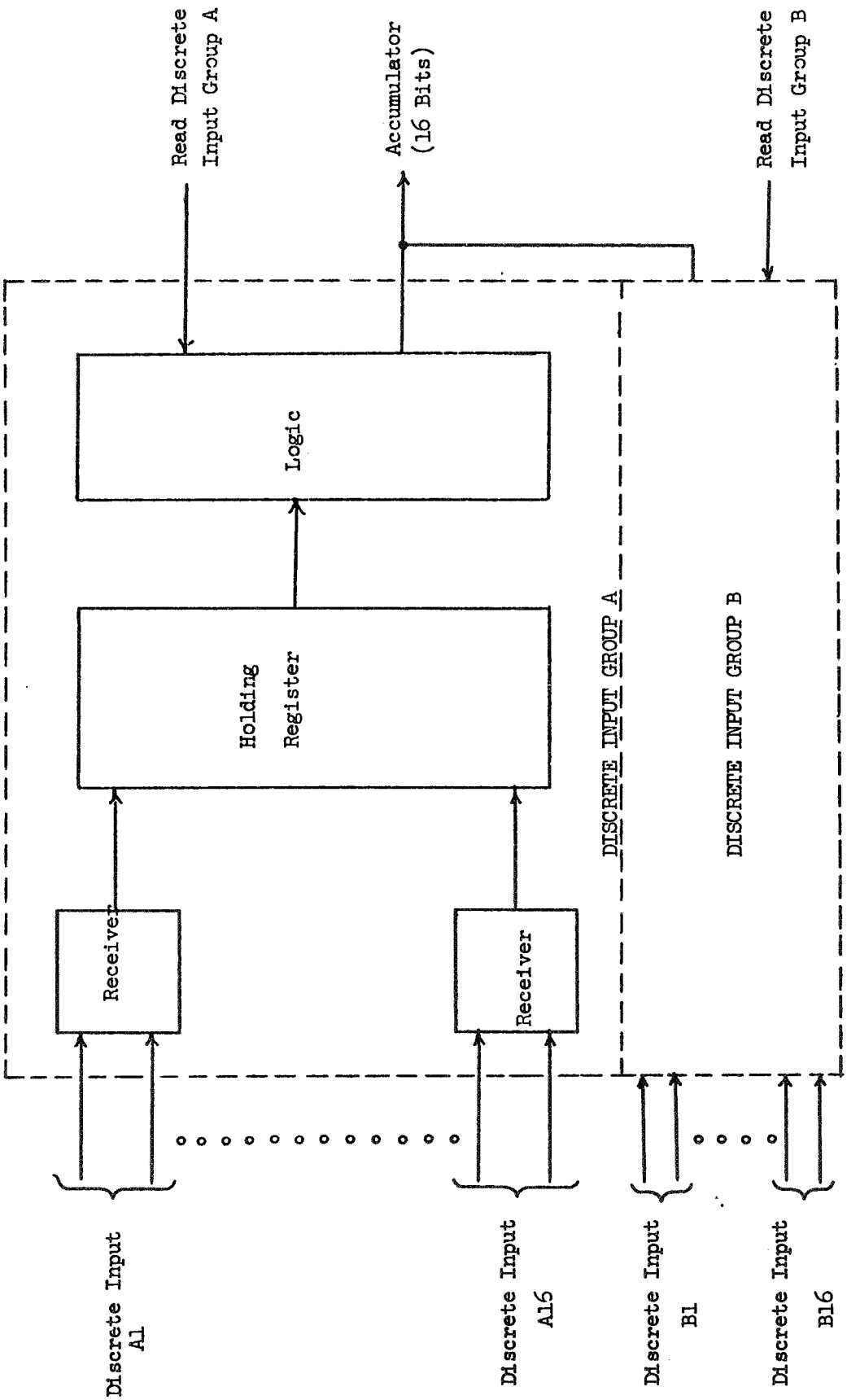


Figure 9-7. Discrete Inputs

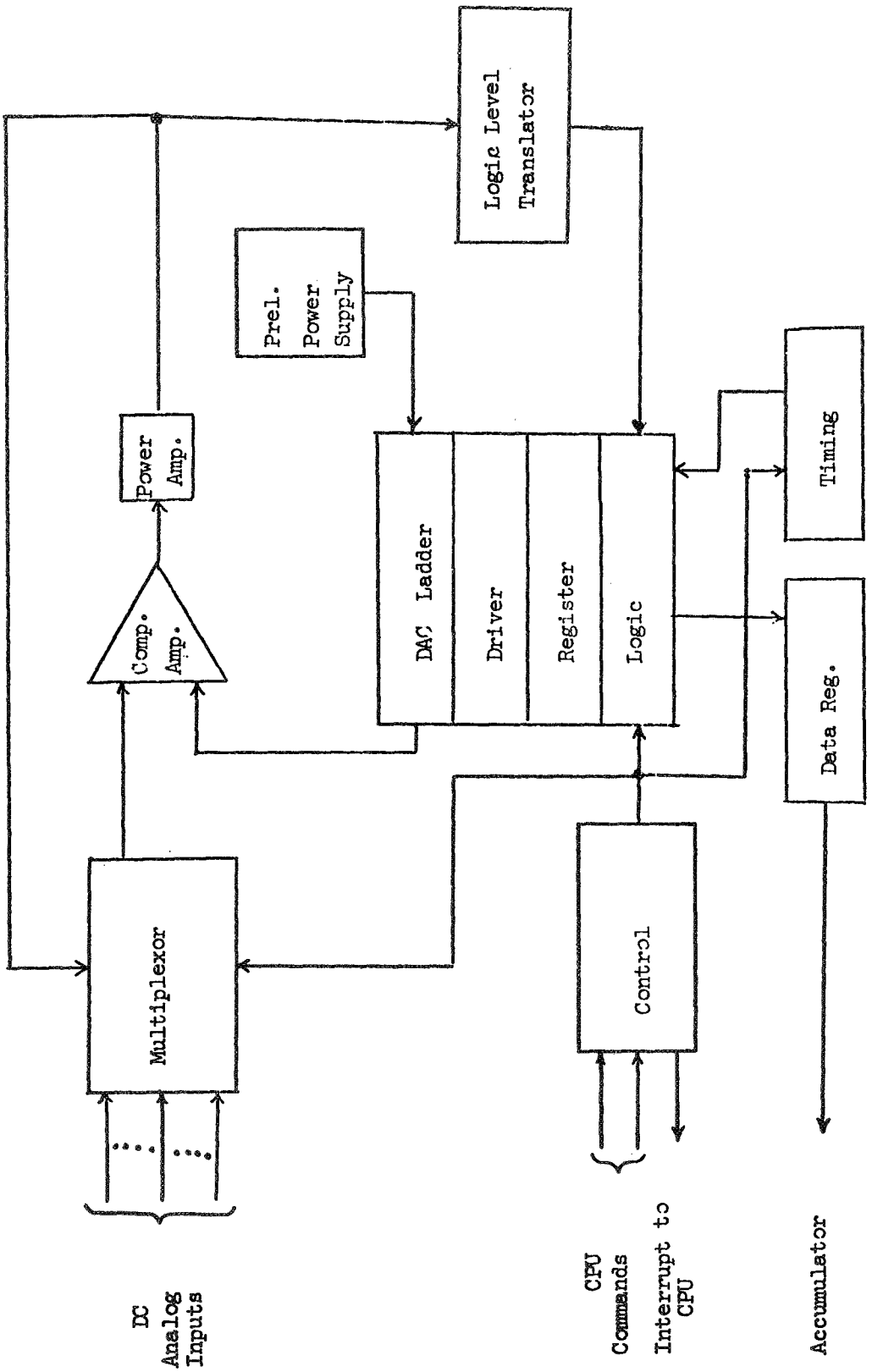


Figure 9-8. Analog to Digital Converter

9.4.3.3 (continued) the converter starts the conversion process on analog input two. A three bit counter that is incremented after each conversion controls the stepping through the inputs. The CPU has 53 microseconds to clear the previous data from the buffer register before the new data is read into the buffer register. This continues until all eight analog inputs have been converted into digital numbers.

9.4.3.4 Parallel Data Bus - The EIU has the capability of sending and receiving parallel digital words over a parallel data bus. The bus handles words in 17 bit format of which 16 bits are data and one bit is parity. Two gating signals to indicate when the data on the data lines are valid are shared by the input and output channels. Figure 9-9 shows a block diagram of the data bus.

Transmissions over the parallel data bus are done on a closed loop basis between the sending and receiving stations. The sending station places the data on line and sets a data valid signal true. The receiving station senses the data valid signal true and reads the data lines into a buffer register. The receiving station now sets a data accepted signal true. The sending station upon receiving the data accepted signal sets the data valid signal false and prepares for the next transmission. When the data valid signal goes false, the receiving station sets the data accepted signal false. The bus is now ready for the next operation.

Transmissions originating outside the LP are started by the receipt of an interrupt by the CPU. The CPU branches to a subroutine which stores enough data for the return to that spot in the main program and then executes an input data word instruction. This causes the parallel data bus control to test the data valid line. When this line goes true, the data lines are read into the buffer register. The data accepted line is set true and the ready signal to the CPU is also set true. The CPU senses the ready line going true and reads the word from the buffer register into the accumulator. The data bus control sets the data accepted signal false when the data valid line goes false. If more words are to be sent, the above operation is repeated with the CPU executing another input data word instruction.

Transmissions originating within the LP start when the CPU sets an output discrete true to alert the receiving station. The CPU then loads the buffer register with the data word and executes the output data word instruction. The outputs of the buffer register are automatically placed on the output channel lines. The data bus control sets the data valid line true and monitors the data accepted line. When the data accepted line goes true, the control sets the data valid line false and sets the ready line to the CPU true. The above is repeated if more words are to be sent.

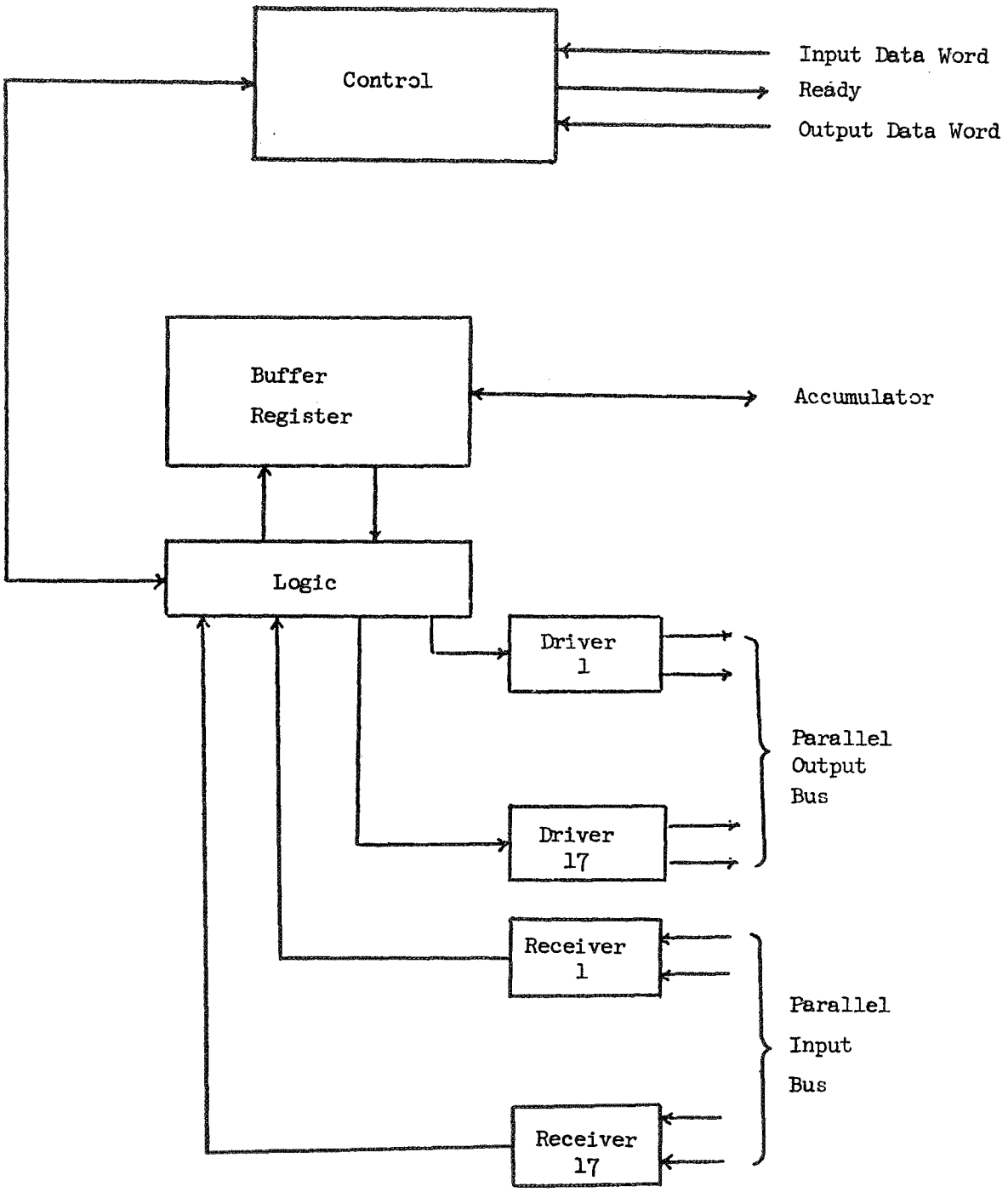


Figure 9-9. Parallel Data Bus

9.4.3.4 (continued)

Single or multiple word messages are handled the same in the EIU. The major difference is the higher effective data rate for the multiple word messages due to a better ratio of data words per channel setup time. The lowest data rate occurs if all messages are single word type. The data rate for this type operation is 66,000 words per second. For multiple word messages, the data rate approaches 143,000 words per second as a limit. In practice, the data rate will be between 66,000 and 100,000 words per second.

9.4.4 Power Converter

The power converter provides the required highly regulated secondary voltages for the LP circuits. The converter operates from a computer grade +28 VDC primary power source. No damage to the power converter or other circuits in the LP will occur due to transients or less of primary power. The LP requires approximately 72 watts of power from the primary source.

9.4.5 Clock

The LP clock used for internal timing is derived from a master oscillator operating at an 8 megahertz frequency. The oscillator is crystal controlled and designed to be relatively insensitive to environmental changes. The oscillator output is used to drive a four phase generator which supplies timing signals to the MOS logic. The oscillator output is counted down to derive a one megahertz timing signal for the bipolar circuits and the memory. The master oscillator has a long term accuracy of ± 100 parts per million.

9.4.6 Physical Characteristics

The LP is contained in a package with the dimensions 7.5 inches high, 13.5 inches wide and 10 inches deep. The LP weighs approximately 18 pounds. All components are mounted on plug-in modules of which there are seven, including the power converter.

9.4.7 Standard Interface Unit

The standard interface unit or SIU provides the interconnection between the I/O Data Bus and the CPU of the Local Processor. It can also be used to interconnect to a subsystem directly if a clock source and memory storage logic are available, i.e., no computational capability is required by the SIU. Each SIU connects to all four I/O bus lines in the G&C System. One output bus is provided from the SIU to the remainder of the Local Processor. The unit has all digital interfaces.

9.4.7 (continued)

Each SIU is independent of all other SIU's and has its own hardwired address. Up to 32 addressable SIU's can be accommodated by the system. All communication between the SIU and the subsystem or the central computer complex is controlled by the central computers. This information is transferred exclusively on the I/O data buses. All data transfers take place at a one megahertz bit rate in a request-acknowledge format.

9.4.7.1 SIU - Data Bus Interconnection - The method of interconnecting an SIU to the four data buses is shown in Figure 9-10. Each data bus is provided with a "T" connection for every Local Processor at the appropriate location along the data link cable. These "T's" have a straight through connection providing one continuous data link cable from end to end. The other part of the "T" contains a tap from each side of the twisted pair line. Fully resistive taps are used, with a resistor in each of the two lines from the twisted pair cable forming a high impedance bridging tap. These can be brought to a connector on the "T" unit or, preferably, connected directly to another twisted pair cable molded directly into the tap structure.

Each tap so formed can be placed up to a few feet from the box housing the SIU. This allows a routing of the four data link cables with spatial separation for damage immunity. The closest any two data link cables need to come to each other is then a function of the box location and the length of the cable from the "T" connection. Utilizing resistive taps directly at the main data link cables reduces the data line degradation due to shorts on the branch cable or at the SIU.

Each branch twisted pair cable is connected to its own connector on the SIU package. Two miniature transformers are connected in parallel to the two wires, one transformer for the data link receiver and one for the transmitter. The transmitters and receivers are individual integrated circuits. There is a filter circuit and bias offset circuit ahead of each receiver to eliminate unwanted signals and noise. Thus each transmitter/receiver pair is AC coupled to the I/O data link bus. Connections at all SIU's are identical. Therefore one transmitter/receiver pair in each SIU is connected in a party line fashion via one data link bus.

Not shown in the interconnection figure are the ends of the four data link bus lines. Each end of each line is terminated in the data link cable characteristic impedance. These terminations occur at the IOP's where the cables originate and end. A similar method of data link interconnect to that outlined in Figure 9-10 is used at the IOP's with only the number of transmitter/receiver pairs differing due to the somewhat different communication interface.

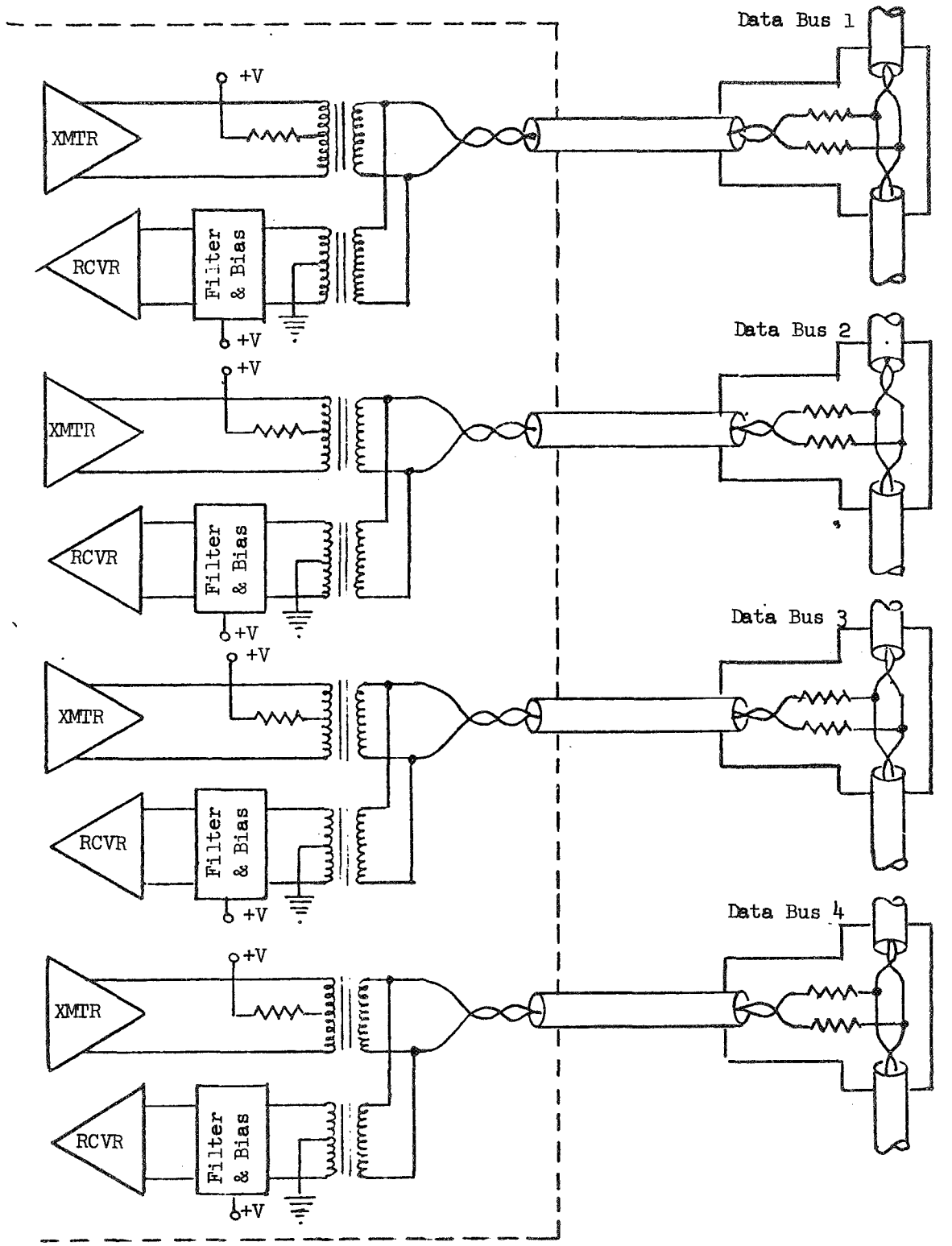


Figure 9-10. SIU - Data Bus Interconnection

9.4.7.2 SIU Operation - Each SIU responds only to communications from one of four sources. These sources are the four IOP's of the central computer complex. Each IOP has a different data bus and is therefore received by onlv one data receiver per SIU. During the normal mode of operation, the SIU is in an idle mode, with its transmitters off and all four of its receivers active. The active receivers are monitoring their individual data buses or IOP's, waiting for a message from the central computer complex.

A block diagram of an SIU is shown in Figure 9-11. Only one of the four receivers is shown, the others are identical. The data transmission sequences for the SIU are discussed in the I/O Data Bus Study, Section 6. Each sequence is initiated by a transmission to the SIU from one IOP. This transmission is received by one of the four receivers. The first and second words received (16 bits each plus parity) are control words and specify the SIU operation for the sequence initiated. For any sequence these control words are identical in format and operation. The word formats are shown in Section 6 (Figure 6-1, page 6-2).

Leading the first control word is a three bit sync code. This is the first information to the SIU receiver and must be detected for proper operation. Proper sync detection starts the receive process and indicates to the individual receiver control that a valid SIU message follows. It also sets the timing and synchronization of all following data for that message.

Every IOP message transmission will be detected by each SIU receiver connected to that IOP data bus. The detection of a correct sync code at the proper time (after no bus transmission or dead band zone) puts the individual SIU receiver into the receive address mode, allowing the input register to be filled from the receiver output. This input register is long enough (16 bits) to hold the first control word that is received. The operation is independent of the mode of the other three receivers.

The first two fields of the first control word specify data to be used by the IOP and are discarded by the SIU. The third field is retransmission data which may be of use to the local processor and is available. The fourth field of the first control word is the local processor address. When this field has entered the shift register it is compared by the address compare circuit to the pre-wired address at each SIU. If the address compares, bit for bit, the SIU-Local Processor that was addressed enters the receive mode, and all others go back to the idle (wait for sync) mode.

The next field is an I/O bit, specifying the message operation of receiving data or transmitting data. Spare bits follow this field and are available for other control functions. Appended to the end of the word is an odd parity bit, bit 17. This is checked against the derived parity for the 16 bits already received. If OK, the SIU receive mode can start its sequence. This causes a CPU interrupt. At this point only one receiver

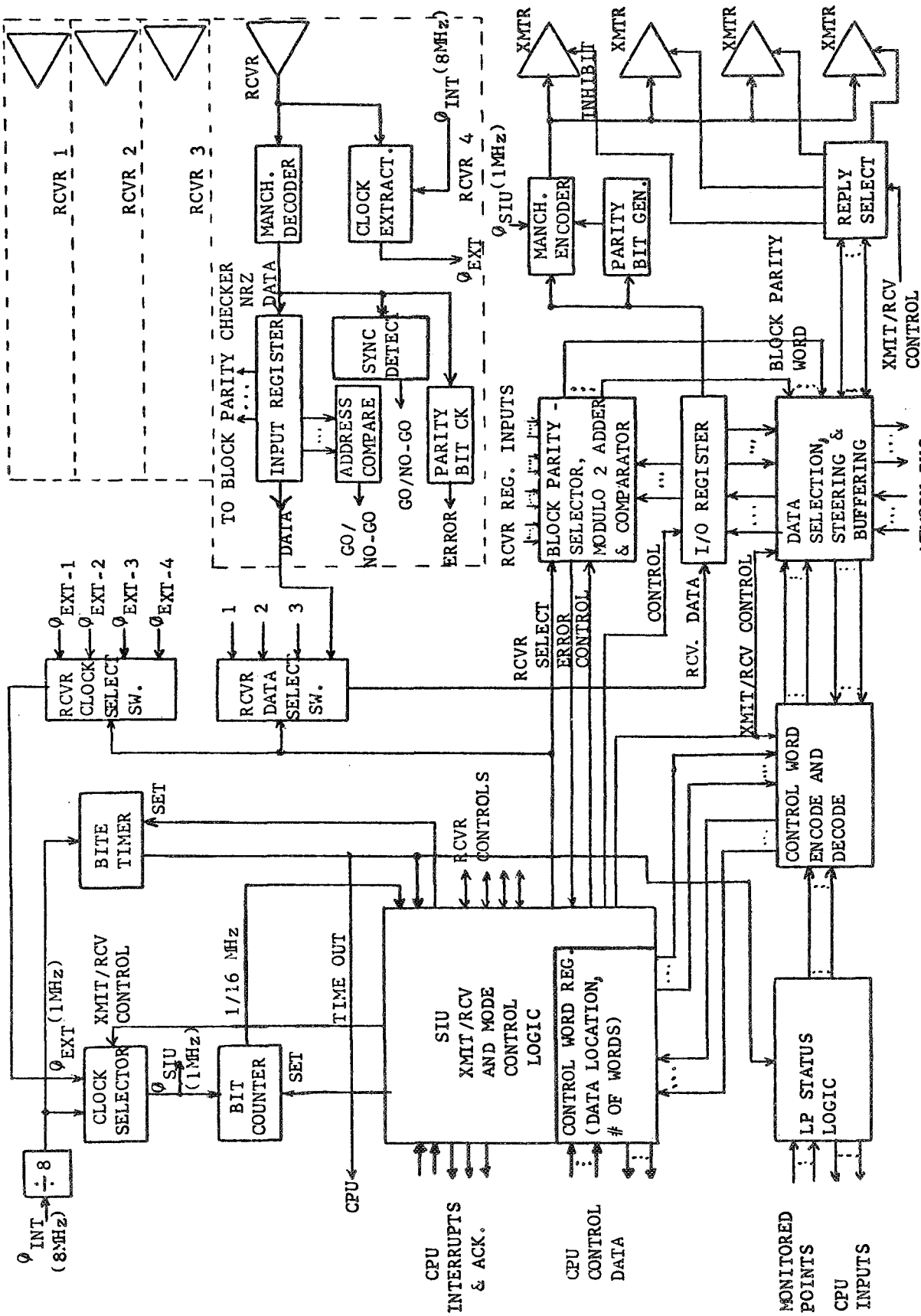


Figure 9-11. SIU Block Diagram

9.4.7.2 (continued) - can be operated and all other receivers are blocked from initiating an SIU receive mode. They still are operational however, and ready to respond on a first-come-first-served basis. The receiver select switches are set for the single receiver being serviced.

In the SIU receive mode, the first control word is shifted into the I/O register as the second control word is received and shifted into the input register. Also the whole 16 bit word is added, module two, by the block parity circuit for later checking. The second control word passes directly through the input register and is followed by one of two types of words. This is indicated by the I/O bit. If the requested operation is to receive data from the IOP, a data word follows. If it is to transmit data, a block parity check word follows.

If a block parity word is to follow, as soon as it has been received serially by the input register, it is compared with the module two sum formed from the first two control words. If it is the complement (odd parity) of this sum no error is present and the SIU can proceed into the transmit mode. If it is incorrect, the SIU returns to the idle mode.

In the other case, where data words follow the control word, the second control word is decoded from the I/O register. This word specifies the data location in memory where the LP CPU has the memory address for the first data word stored, and the number of words in the message. The number of words field is stored by the CPU and used to determine the length of the ensuing message. After each data word is in the I/O register, it is transferred to the buffer register and a CPU interrupt generated. The data word is stored under CPU program control into the proper memory location during the next 15 μsec. As the last data word fills the I/O register, the block parity word for the whole message fills the input register. (All words in the selected input register are added module two for a message). This is compared for a receive data error. Errors are reported to the CPU in the SIU acknowledge. The CPU determines the end of the message and signals the end of the receive sequence (this is checked) and the SIU enters the transmit mode as before.

Two sequences exist for the transmit mode, one in which an acknowledge is transmitted, and the second in which data is transmitted. The transmit acknowledge takes place after the data receive mode ends. Two acknowledge words are transmitted by the SIU, similar to the two control words. The transmitters used for this operation were specified by the second field of the second control word, and can be one, two, three or all four.

9.4.7.2 (continued)

The first acknowledge word contains a field for the reply bus information (output of Reply Select Circuit), the type field is repeated, the data location register contents are placed in the data location field and the I/O bit set for the operation the SIU is performing. The second acknowledge word contains a field for the local processor address (pre-wired), the number of words just received or to be transmitted, and the status of the LP.

This data is prefaced by a pre-wired sync code of three bits for SIU-IOP communication. Then the two acknowledge words plus a parity bit for each are transmitted. The block parity check sum is inserted in the I/O register and serially transmitted last. The SIU returns to the idle mode.

Theother transmit sequence is similar to the data receive mode, except the second acknowledge word is followed by data. This data was specified by the received second control word and is read out of (rather than into) the memory a word at a time under CPU control into the data buffer register. It is parallel transferred into the I/O register at the start of each transmit word time. All transmitted words are added modulo two so the check sum can be entered and transmitted after the last data word.

Transmissions by an SIU will be ignored by its own receivers and all others except the IOP's due to the non-detectable sync code preceding each SIU transmission. During the receive mode the clock source for all timing is derived from the received data stream. In this fashion all data is decoded and distributed under control of the clock used to assemble it. Each individual receiver control uses its own line derived clock until the SIU enters the receive mode and all other receivers are locked out. When the SIU is transmitting it uses the LP 8 Mhz clock divided down to 1 Mhz for all internal clocking and Manchester encoding.

A BITE timer is used to monitor all data transmissions and receptions. Since these messages are all of a fixed maximum length this timer can signal the necessity to lock out a continually operating receiver or turn off the SIU transmitters upon SIU failures. This action is reported to the CPU and to the IOP through the LP status word when possible.

10.0 POWER DISTRIBUTION INVESTIGATION

10.1 INTRODUCTION

This section discusses the Power Distribution Study. The objective was to determine a preferred power distribution system for the G&C system. The preferred system, described in Section 10.7, consists of solid state load controllers packaged with the G&C load at the interface point with the Electrical Power Subsystem (EPS). The load controllers are controlled by a logic level power control module in the Local Processor (LP). The power to the LP is controlled separately to afford a method of isolating failed LP's by removing power.

The first discussion concerns characteristics of Mil-Std-704A input power and its impact on power converter design. It is interesting to note that special "computer grade" power has been proposed on some recent avionic systems to circumvent the design problems introduced by Mil-Std-704A. Next, the generalized characteristics for a distribution system are developed along with failure characteristics. Various types of load and source isolators are discussed in Appendix 8.

10.2 INTERFACE CHARACTERISTICS

10.2.1 Input Power

Mil-Std-704A power has been assumed as the power available at the G&C system interface for the purposes of this study (Ref. 10-1, 6.5-3.2). Less severe transient surge voltages defined by curves 5 and 6 of Mil-Std-704A have also been considered (Ref. 10-2). For convenience, the characteristics of this power is summarized in Tables 10-1 and 10-2.

TABLE 10-1

Summary of Mil-Std-704A, Category B-AC Power, Nominal 3Ø, 400 Hz, 115 VAC.

	NSSL	ASSL	ESSL	Full Transient	Limited Transient
Voltage Min *	108	102	104	58	74
(Volts) Max	118	124	122	180	140
Frequency Min	380	-	360	320	370
(Hertz) Max	420	-	440	480	430

* Min. voltage for space station may be 1.5 volts greater due to less maximum voltage drop in distribution bus (Ref. 10-1).

10.2.1 (continued)

TABLE 10-2

Summary of Mil-Std-704A, Category B-DC Power, Nominal 28 VDC					
	NSSL	ASSL	ESSL	Engine Start	Landing
Voltage Min (Volts)	24	22.5	16	16	22.5
Max	28.5	30.0	24	28.5	28.5
Transient Voltages (Volts)	Full Transient	Limited Transients		Spike	
Min	8	12		- 600	
Max	80	60		+ 600	

NSSL - Normal Steady State Limit
ASSL - Abnormal Steady State Limit
ESSL - Emergency Steady State Limit

Mil-Std-704A AC input power has the advantage of being well defined and of having characteristics well known to designers of avionic equipments. Much equipment already exists that operates from it and therefore it presents low technical risk and lower cost for equipment design or modification.

However, since the volume and weight of magnetic components varies inversely with the three-fourths power of frequency (Ref. 10-3), there is a size and weight advantage of going to a higher frequency source. The main disadvantage is that less equipment has been designed for higher frequency AC power and development cost and risks could be anticipated to be greater. Direct distribution of this higher frequency power is desirable. However, it can be generated in the load if necessary or desirable.

One method used to reduce the volume and weight of magnetics in some electronic equipment using Mil-Std-704A power is to use the DC power and static DC to DC converters operating at frequencies determined by the switching limitations of the solid state switch and the required volt/turn resolution of the transformer.

10.2.1 (continued)

For transistor switches, conversion frequencies to 5 KHz are usual, to 20 KHz common, and to 100 KHz or higher possible. Not only are the power transformers reduced in volume and weight, but the required filtering is greatly reduced at these frequencies. The reduced energy storage of the reduced filters essentially turns the transient voltages of Mil-Std-704A into steady state voltage and regulation is usually required. The eight to eighty volt swing makes dissipative regulator inefficient. Since the open loop gain of many types of switching regulators is proportional to the input voltage, it makes switching regulators more difficult to stabilize. Also selecting the power transistor for switching regulators is a problem. High breakdown voltage is required at the 80V limit and high current capability at the 8 volt limit. This combined with fast switching times and adequate forward and reverse secondary breakdown ratings makes the power transistor design a risky, expensive, low yield compromise of conflicting requirements. There is much to be gained if the 28 VDC bus can be held within more reasonable limits, say 24 to 32 volts. This would allow the equipment to meet the audio susceptibility requirements of Mil-I-6181D or Mil-Std-461A with the input at 28 volts and work from the normal steady state level of 24 to 28.5 V of Mil-Std-704A.

Static inverters are a source of radio frequency noise and line filters used to filter this noise can usually be designed to sufficiently attenuate very short duration spikes (a few microseconds) outside the normal steady state value.

Another method used to reduce the volume and weight of magnetics operating from Mil-Std-704A power is to directly rectify the AC power and operate a DC to DC converter from the resulting high voltage DC. This is possible because of the availability of power transistors with very high voltage breakdown. Although the components and design techniques needed for this type conversion are different, the arguments are the same for limiting the transient tolerance range.

10.2.2 Load Power

Power controllers in the G&C system will have to handle maximum loads from 10 watts to over 2 kilowatts (Ref. 10-1). These are summarized in Table 10-3. Power controllers within the computer may be in the low watt range. For the purpose of this study, load controllers over a range of 28W to 8KW and 1.0 to 75 amperes are within the range to be considered. Power may be DC, one phase AC or three phase AC.

10.2.3 Environment

Discussion of the environmental characteristics of the various components in the G&C power distribution system will include only thermal considerations since the behavior of solid state power devices to other environmental conditions should be similar to other solid state components in the system.

TABLE 10-3

SUMMARY OF REQUIREMENTS

					Artificial Environment		
	Ave	Max	Emerg	Launch	Ave	Max	Emerg
CMG's	450	1360	460	460	460	1360	460
Tracker	200	200					
Horizon Edge Tracker	10	10	10	10	10	10	10
IMU	160	200	160	160	160	200	160
G&N Computer	20	20	20	20	20	20	20
Docking Sensor		200				200	
Miscellaneous	30	35	30	30	30	35	30
TOTAL	880*	2075	680	680	680	1825	680

* 10 watt positive adding error in reference copied (Ref. 10-1)

10.3 GENERALIZED DISTRIBUTION BUS SYSTEM

10.3.1 DC Power

Figure 10-1 shows a distribution system made up of two sources, two busses, and two loads. The sources and loads are connected to the busses with isolators. Each bus needs one isolator for each source and one isolator for each load that is to be connected to it. For example, four sources, four busses and sixteen loads would require sixteen source isolators and sixty-four load isolators or eighty isolators.

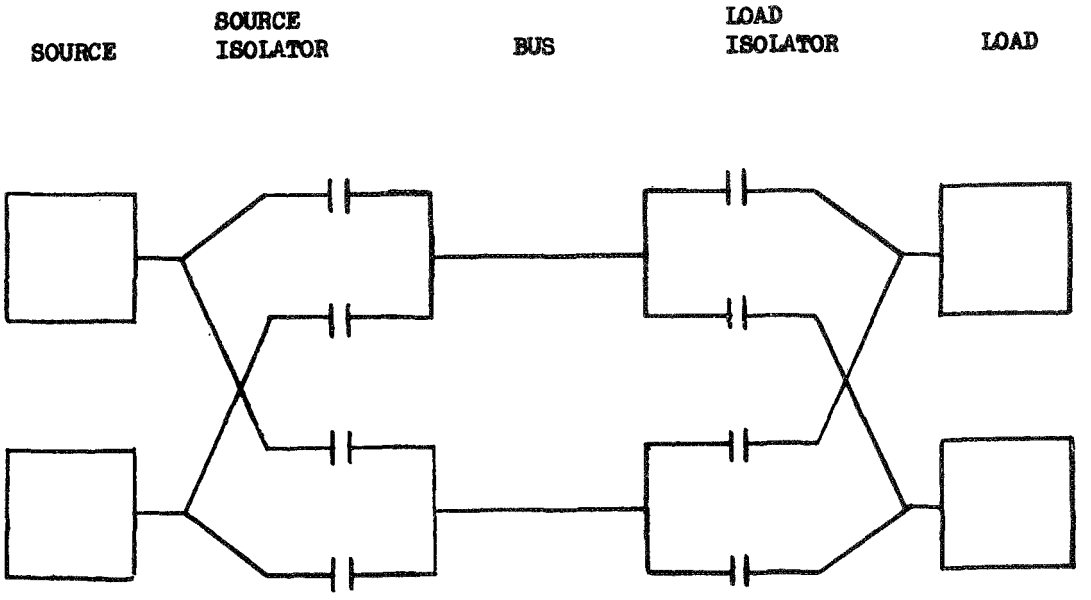


FIGURE 10-1. GENERALIZED DISTRIBUTION SYSTEM

10.3.1 (continued)

The necessary characteristics of the isolator depend on the possible failure modes of the source or load.

Tables 10-4 and 10-5 list the various isolator characteristics for various failure modes of DC sources and loads. Two things can be observed from the data in the tables. First is that the only isolator that works for all failure modes is a series switch. Since it isolates a failure by disconnecting either the source or power, the failure mode can never be fail operational. If the series switch fails after a load or source failure, it must always fail open in order to maintain isolation. If the FOOS criteria means that power must be available after three failures, then four sources and four busses are required to meet the criteria.

The second observation that can be made from the tables is that if the source or load have certain limited possible failures modes, then other isolators are available that may let the failure criteria be met with something less than brute force quadruple redundancy.

It is of interest to note that the results shown in the tables are valid for other than a power interface. For example, the source could be a digital driver and the load its receiver.

10.3.2 AC Power

In concept, it would be possible to develop a failure matrix for an AC system similar to the one developed for the DC system. Additional failure modes such as low frequency, high frequency and phase rotation reversal would need to be considered. However, the general conclusions would be the same. The only universal isolator is a series switch that fails open; the failure mode is fail safe, and quadruple redundancy is needed to meet the requirements. If failure modes of the sources or loads are limited, it may be possible to have fail Op failures and reduce the order of redundancy. One thing that should be mentioned is that by using a four-wire wye input for three phase transformers, they can be designed to be fail Op if any single wire opens.

In addition to circuit breakers, magnetic components such as saturable reactors and others can be used as series switches.

TABLE 10-4

DC SOURCE FAILURE MATRIX

Source, Failure Mode		Isolator	Failure Result At Bus	Required Isolator Failure Mode	Example Device or Circuit
Voltage	Impedance				
High	High	Shunt VR	(1)	Short (2)	Zener Diode
		Series VR	(1)	Open	Series Voltage Regulator
		Shunt Switch	(2)	Short (2)	4-Layer Diode
		Series Switch	Safe	Open	Circuit Breaker
High	Low	Series VR	OP	Open	Series Voltage Regulator
		Series Switch	Safe	Open	Circuit Breaker
Low	High	None	Safe (3)	-	-
		Blocking Diode	Safe	Open Short (3)	Diode
		Series Switch	Safe	Open Short (3)	Circuit Breaker
Low	Low	Blocking Diode	Safe	Open	Diode
		Series Switch	Safe	Open	Circuit Breaker
Reversed	-	Blocking Diode	Safe	Open	Diode
		Series Switch	Safe	Open	Circuit Breaker

(1) Safe or Operational Depending on Impedance
(2) Converts to Low-Low Failure Mode
(3) Impedance of Failed Source Loads Redundant Bus
VR = Voltage Regulator
OP = Operational

TABLE 10-5

DC LOAD FAILURE MATRIX

Load Failure Mode		Isolator	Failure Result At Bus	Required Isolator Failure Mode	Example Device or Circuit
Voltage	Impedance				
Ground	Low	Current Limiter	OP	Open	Resistor, Current Regulator Diode
		Series Switch	OP	Open	Circuit Breaker
Positive	Low	Blocking Diode		Open	Diode
		Current Limiter	(1)	Open	Resistor, Current Regulator Diode
		Series Switch	OP	Open	Circuit Breaker
Negative	Low	Current Limiter	OP	Open	Resistor, Current Regulator Diode
		Series Switch	OP	Open	Circuit Breaker

(1) Current Fed Back On To Bus Which Could Fail Bus

OP = Operational

10.4 LOAD ISOLATOR FAILURE CHARACTERISTICS

It is assumed that the source isolators are part of the Space Station power distribution system and that four busses are made available to the G&C system load. This would be the most complex system and can be easily simplified. These busses may be handled by several combinations of switching and oring as shown in Table 10-6. The dotted lines indicate some form of oring inside the load. An example would be a transformer rectifier on each 3Ø AC bus with the DC outputs tied together. The series switch isolators are all considered to be break before make. This is necessary for AC power from sources not in sync and may be desirable in any event to prevent tying a good bus into a failed bus. The disadvantage is that the load is without power during the break before make period.

The normal switch logic is for normal power-up operation. Different logic may be desirable after failure detection, during bus maintenance or checkout, or for degraded operation modes.

If the assumption is made that oring in the load requires additional hardware, which is probably a good assumption, then Configuration 2 is the best for loads that can tolerate a Break Before Make (BBM) transient loss of power. If BBM transients are unacceptable, Configuration 3, 5, or 6 is necessary with 3 being the least complex.

Where the statement "can be fail OP. . . ." in the "Bus Fail Short" column is made, it means that this is possible for isolators with proper characteristics, namely, unilateral power flow.

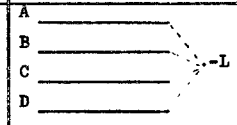
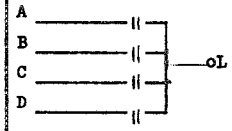
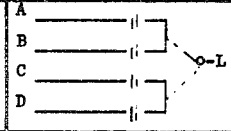
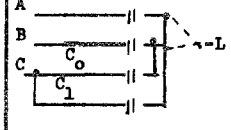
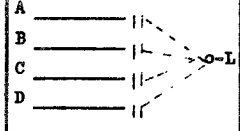
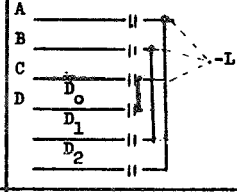
It should be noted that a shorted load may pull the bus down before the isolator disconnects it. This can be prevented if either the isolator or load contains current limiting.

10.5 EXAMPLES OF BUS SWITCHING

Figure 10-2 is an example of Configuration 2 of Table 10-6 where one of four busses may be switched to the load with break before make switches causing a momentary loss of power at the load. Figure 10-3 is an example of Configuration 3 of the same table where two busses are or-ed in the load and each bus is backed up by a redundant bus connected with break before make switches. In this configuration the load sees no momentary loss of power. The two configurations are quite similar in hardware and failure modes, the latter requiring additional complexity in the load and additional monitors and voters.

The power switch is assumed to be a series switch that trips open if overloaded. It is turned on or off as commanded by control line. It can be reset after tripping for an overload in some way. This could be by cycling the input logic line off and on, by removing input power or by use of a separate control line.

BUS/ISOLATOR FAILURE CHARACTERISTICS MATRIX

	Bus/Isolator	Normal Switch Logic	Effect on Bus for Shorted Load	EFFECT ON LOAD FOR BUS FAILURE		Isolator Failure		Isolator Open Any Load	Logic Failure
				Fail Open	Fail Short	Isolator Short: Shorted Load	Isolator Short: Normal Load		
1		L = A + B + C + D	Fails Four Busses (Short)	Fail OP three times with no BBM* transient	Can be fail OP three times with no BBM transient	Isolator Undefined			Not applicable
2		L = $A\bar{B}\bar{C}\bar{D}$ + $\bar{A}B\bar{C}\bar{D}$ + $\bar{A}\bar{B}C\bar{D}$ + $\bar{A}\bar{B}\bar{C}D$	Bus Protected by Isolator	Fail OP three times with BBM transient	Fail OP three times with BBM transient	Loss of Bus. Effect on other busses depends on logic and isolator characteristics	Can not turn off load. Effect on other busses depends on logic and isolator characteristics	Fail OP three times with BBM transient	Can not turn load off and/or on. If two or more isolators are on effect depends on isolator characteristics
3		L = $A\bar{B}\bar{C}\bar{D}$ + $A\bar{B}C\bar{D}$ + $\bar{A}B\bar{C}\bar{D}$ + $\bar{A}B\bar{C}D$	Bus Protected by Isolator	Fail OP three times with no BBM transient	Can be fail OP three times with no BBM transient	Same as 2	Same as 2	Fail OP three times, at least once with no BBM transient	Same as 2 except more probability of being able to turn load on
4		L = $AB\bar{C}_o\bar{C}_1$ + $A\bar{B}C_o\bar{C}_1$ + $\bar{A}B\bar{C}_oC_1$	Bus Protected by Isolator	Fail OP two times with no BBM transient	Can be fail OP two times with no BBM transient	Same as 2	Same as 2	Fail OP two times with no BBM transient	Same as 2
5		L = A + B + C + D	Bus Protected by Isolator	Same as 1	Same as 1	Same as 2	Same as 2	Same as 4	Same as 2
6		L = $AB\bar{C}\bar{D}_o\bar{D}_1\bar{D}_2$ + $AB\bar{C}D_o\bar{D}_1\bar{D}_2$ + $A\bar{B}C\bar{D}_oD_1\bar{D}_2$ + $\bar{A}BC\bar{D}_o\bar{D}_1D_2$	Bus Protected by Isolator	Fail OP three times with no BBM transient	Same as 1	Same as 2	Same as 2	Fail OP three times with no BBM transient	Same as 2

*BBM = Break Before Make

TABLE 10-6

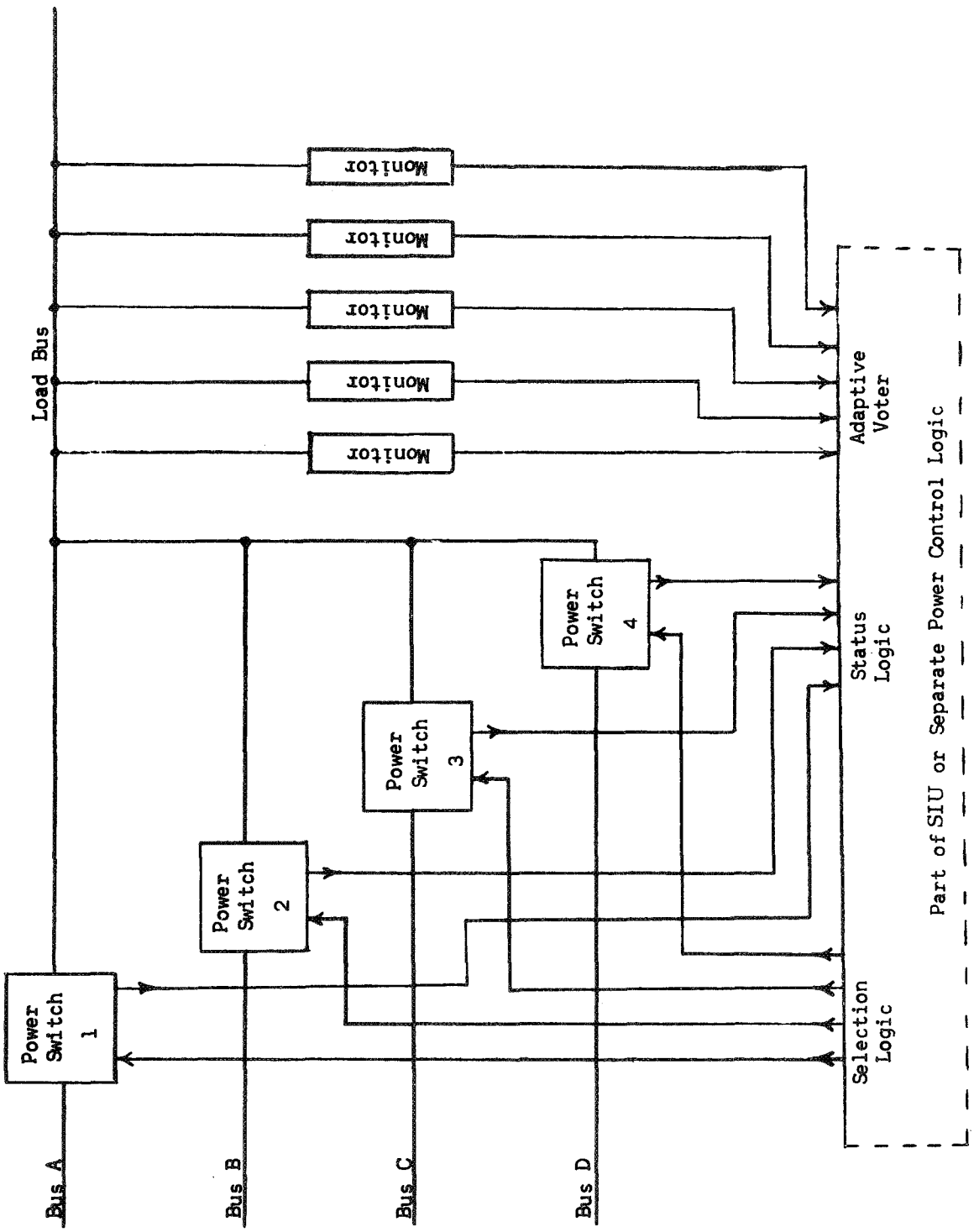


Figure 10-2 BUS SWITCHING CONFIGURATION 2

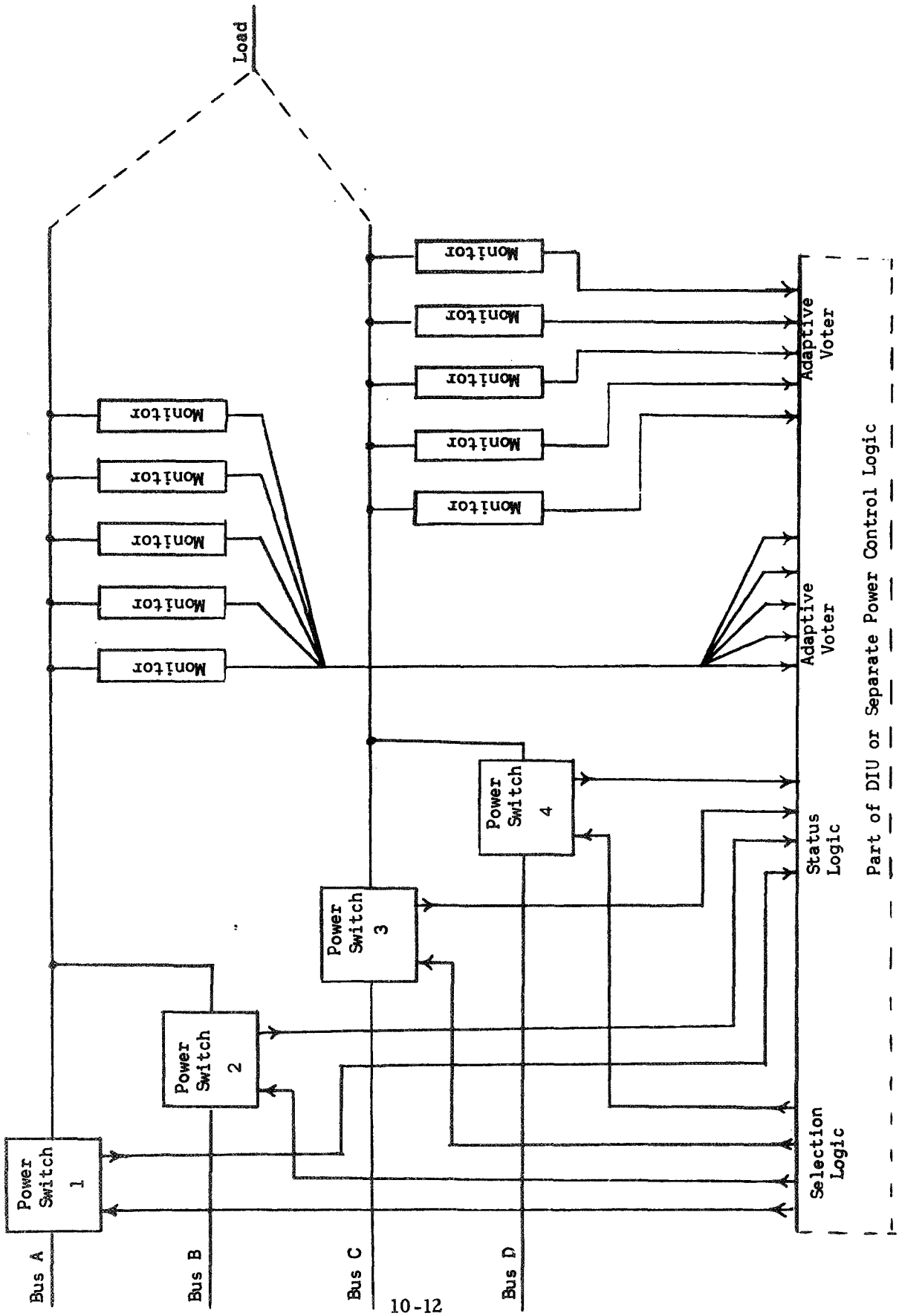


Figure 10-3. BUS SWITCHING CONFIGURATION 3

10.5 (continued)

As a minimum, the status line indicates what the power switch thinks it sees on the control line. It is probably desirable that the status line indicates if the switch has tripped because of an overload.

The examples operate as follows:

10.5.1 Checkout

1. Command Power OFF
2. Check status lines, adaptive voter output and individual monitor outputs.
3. Initiate failure isolation sequence if false, proceed if true.
4. Sequence each switch on in turn and check status lines, adaptive voter output and individual monitor output.
5. Initiate failure isolation sequence if false, declare system operational if true.

10.5.2 Normal Operation

Bus selection may be by priority logic, pre-programmed logic or random depending on selection logic. In any event power is programmed on or off as commanded.

10.5.3 Failure Mode Operation - Power Loss

When adaptive voter output indicates loss of power, the active bus is switched off and a new bus switched on. There is a loss of power transient. This can be repeated until all busses have been tried and then terminated or cycled continuously.

10.5.4 Failure Mode Operation - Shorted Load

For a true short it is desirable to declare the load knocked out and go into some system backup mode. This is desirable because only one or two busses see an overload stress. The disadvantage is that if the overload indicator is in error and the load is not shorted, the system has been disabled by one failure. For this reason it seems desirable to try to drive a short with at least one more bus to verify that the load is truly shorted. The extreme would be to cycle the shorted load onto each bus at least once.

10.5.5 Failure Mode Operation - Non-Commanded Power Application

This may be an impossible or an acceptable failure mode. If not, a backup fail safe element must be included that can be dependably opened. This might be a backup switch that is part of the main power distribution

10.5.5 (continued) system or a redundant independently activated switch or fuse link in the power switch. Deciding which switch is the failed one may be a problem, especially if the result of activating the redundant switch is irreversible such as blowing a fuse link.

10.5.6 Failure Mode Operation - Monitor Failure

The adaptive voter is assumed to have the capability of detecting when one monitor is reading different from the majority and masking its output from future decisions. By using five monitors and a five input voter, three monitor failures still produce a valid output from the voter. Given additional knowledge from the system such as the selection and status commands and prior history, fewer monitors may be necessary for triple fail-op.

10.5.7 Failure Mode Operation - Logic Failure

As a minimum, logic failure can induce failures that appear as power loss and non-commanded power application failures. These can be protected against and the ability to simulate the failures logically may be a desirable checkout tool. Other logic failures could prevent meeting the triple fail-op criteria and therefore be unacceptable. Circumventing these unacceptable failure modes is a major consideration in implementing the logic.

10.6 COMPARISON OF ISOLATION DEVICES

An investigation of various isolation devices and their failure modes was performed during the study. Diodes for DC applications include computer diodes, diffused diodes, ion implantation diodes and Schotty barrier hot carrier diodes. For AC systems, magnetic isolators of three types were investigated. These were a simple magnetic amplifier, a four aperture magnetic switch developed by Stanford Research Institute for Jet Propulsion Laboratory under NASA contract, and the parametric magnetic device called Paraformer™, a product of Wanlass Electric Co. Also, solid state controllers, under development for aircraft applications were examined, and finally, electromechanical relays were considered briefly for comparison purposes. Detailed information on the operation and failure modes of these devices is presented in Appendix 8.

Table 10-7 compares various characteristics of AC isolators. The transformer is given as a reference for the magnetic devices. The volume and weight efficiencies of the solid state and conventional circuit breakers are around two orders of magnitude better than the magnetic devices. The solid state device has as good efficiency as the magnetic devices, but is poorer than the nearly lossless conventional circuit breaker. However, the solid state device presents the greatest cooling problem, dissipating

10.6 (continued) 32 times more power per cubic inch than a transformer with a 40°C temperature rise, and ten times more than a conventional breaker. This, combined with the sensitivity of reliability to temperature of solid state circuitry, makes cooling of these devices critical.

TABLE 10-7

COMPARISON OF AC ISOLATOR DEVICES

	Transformer (1)	Parametric Transformer (2)	SRI* Switch (3)	Magnetic Amplifier (4)	Solid State Controller (5)	Circuit Breaker (6)
Power	100w	100w	40w	100w	1150w	2300w
Frequency	400 Hz	400 Hz	2.5 KHz	400 Hz	400 Hz	400 Hz
Dimensions	2.3 x 2.1 x 3.1 in.		6.5 x 1.6 x 2.3 in.	3.2 dia. x 4.5 in.	1 x 1 x 1 in.	1.5 x 0.6 x 1.6 in.
Volume	15.0 in^3	45 in^3	25 in^3	35 in^3	1 in^3	1.5 in^3
w/in^3	6.7	2.2	1.6	2.88	1150	1530
Weight	1.9 lb.	4.7 lb.	-	2 lbs.	0.13 lb.	0.13 lb.
w/lb.	53	21	--	50	8000	16000
Power loss	7.5w	15w	1.6w	19w	16w	2.1w
efficiency	93%	87%	96%	84%	88%	99%
Power Loss/in^3	0.5	0.33	0.063	0.54	16w	1.4w
Remarks	Comparison only, not used as switch	Also provides voltage regulation	2.5KHz Square wave input power		Smallest size unit	Smallest size unit

* Different frequency, power level, and leaving out drive transformer and control power makes comparison with others invalid.

(1) Ref. (10-4) Chapter 5. Assumes 40°c Rise, 3.5% regulation and equal core and coil losses.

(2) Volume estimated at 3X, and weight as 2.5X, transformer.

(3) Ref. (10-5).

(4) Ref. (10-6). Includes control power and inductor in current source.

(5) Ref. (10-7).

(6) Ref. (10-8). -40°c to +100°c temperature range.

10.7 RECOMMENDED CONFIGURATION

The recommended configuration is based upon the following constraints suggested by the general system studies:

a) When power is supplied from the Electrical Power Subsystem, all/most of the LP's and G&C subsystems will remain powered-down. Sufficient circuitry will be energized to recognize a LP address and LP power on/off command from the data bus or an auxilliary bus.

b) Power can be applied to and removed from any LP by commands on the data bus and/or an auxilliary bus.

c) Subsystem power is controlled by a powered-up LP.

Figure 10-4 is a block diagram of a configuration meeting these constraints. The various blocks are discussed in detail and then specific recommendations with rationale are given.

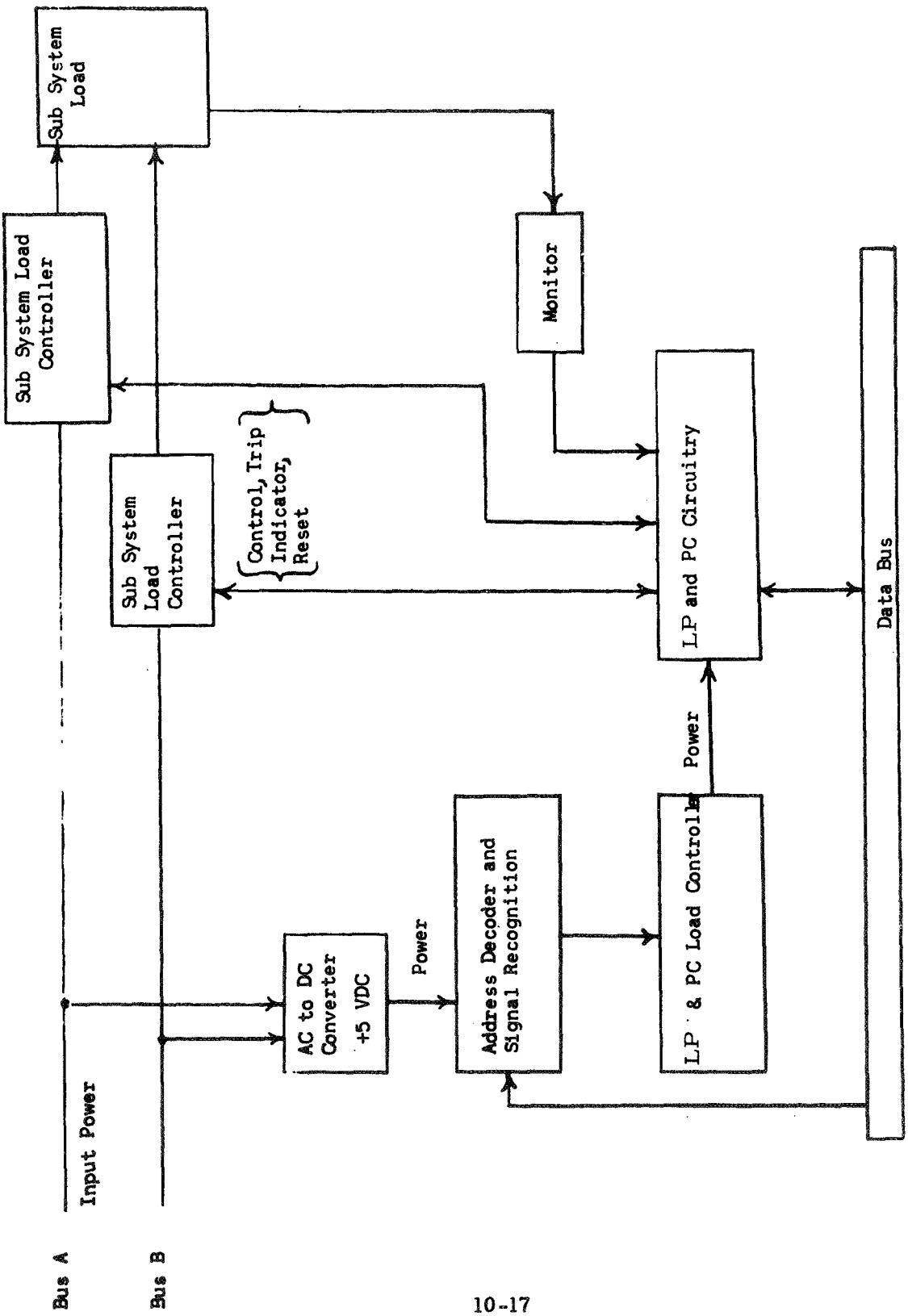
10.7.1 Subsystem Load Controllers

Characteristics are assumed to be similar to those listed in Reference (10-11) and discussed in Appendix 8. The internal configuration should be such that any internal short that would load the power input would activate the fail safe current circuitry assuring a fail open failure mode. The AC load controllers are bilateral devices with undefined failure modes if one bus is tied to the input and another to the output. Since this is a possible condition if there is a logic or load controller failure, the failure mode should be an acceptable and well defined one. Controllers for three-phase power are tied together through the power control logic so that they turn on, off, and trip together. As indicated by the power loss/in^3 row of Table 10-7, the location of the part will be heavily influenced by thermal design requirements.

The load controllers should use EMI suppression techniques such as zero crossover switching for AC controllers and controlled waveforms for DC controllersto generate acceptably small EMI. Since normal design practice is to place EMI filters right at, or as part of, the input power connectors, interposing a load controller between the input connector and filter could compound EMI control problems for the individual subsystem designers.

10.7.2 LP and PC Circuitry

The Power Control (PC) logic can be part of the LP, part of the load controller, or a separate module. If part of the load controller or LP, it can share the power supplies and packaging of these units which



Power Control Block Diagram
Figure 10-4

10.7.2 (continued) is a definite advantage if the added complexity in the LP or load controller are acceptable. Figure 10-5 is a flow diagram indicating what this logic might look like. The logic could be hardwired, under software control, or a combination. Some options may be left out--for example, the option to try a new bus if the trip indicator indicates a shorted load.

10.7.3 Monitor

The monitor circuit is a simple circuit that detects the presence or absence of acceptable power. It can be located at the power controllers, with the power control logic, or at the subsystem load. If located with the load, it provides the most credible information about the presence of power at the load.

10.7.4 LP and PC Load Controller

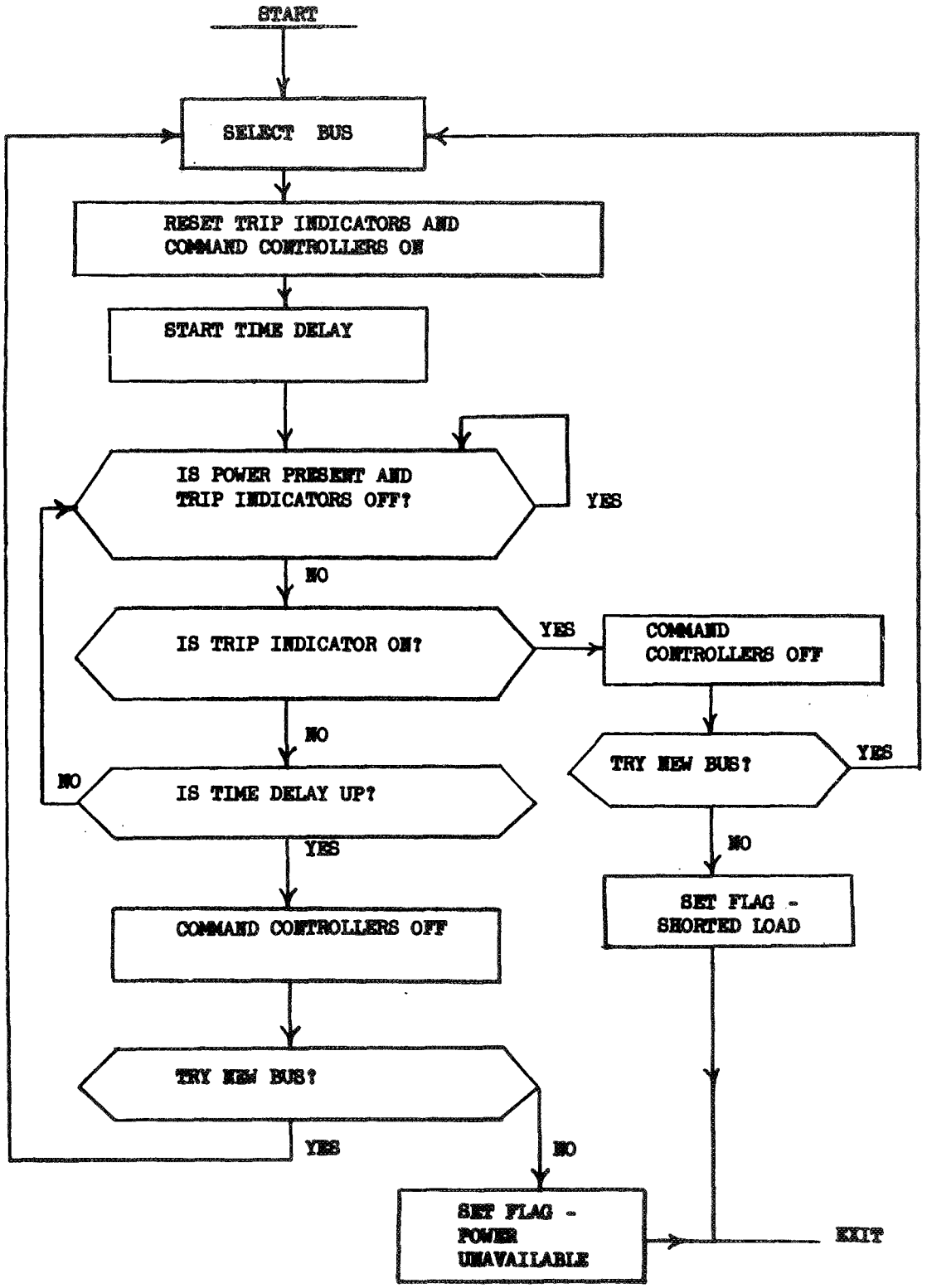
The purpose of this controller is to conserve power when a LP and subsystem are not activated and to isolate a failed LP. The physical placement of this controller and/or the associated address decoder and signal recognition circuitry may be with the LP or physically separated from it, depending on how a failed LP is to be isolated in the overall system.

10.7.5 Specific Recommendations and Rationale

10.7.5.1 - Use solid state load controllers similar to those described but with additional considerations given to failure modes. Rationale: These devices are competitive with mechanical breaks in all aspects except power loss and offer orders of magnitude improvement in number of operating cycles. (Table 10-7). The technology is in a high rate of development for aircraft electrical systems and should not present any development risk for the Space Station program.

10.7.5.2 - Package the controllers as part of the load electronics if thermal and EMI design permits. Each bus should be brought into the load controller on a separate connector, which is the interface with the electrical power subsystem. It is assumed each bus will be brought to the load in a separately routed cable. Rationale: Simple interface and consistent with EMI and mechanical redundancy design practice.

10.7.5.3 - Package the power control logic as a separate module in the LP. Communicate with the load controllers on the normal LP/Load interface. If LP loses power, load controller must open. Logic will be hardwired. Rationale: Logic can be powered from LP internal power. Simple interface with data bus and load. It is desirable to have power control identical for all loads but if not, modules with appropriate logic



POWER CONTROL LOGIC FLOW DIAGRAM

FIGURE 10-5

10.7.5.3 - (continued) - can be plugged into LP. This allows failed LP to be isolated by removing power. Software controlled logic offers only minor advantages in trouble shooting and operational flexibility.

10.7.5.4 - Power LP with separate load controller. The location of this controller and its logic are to be determined by system considerations to be made later from the following options.

a) Package as part of LP and control through data bus.

b) Package separately from controlled LP (may be in one or more separate LP's) and control through data bus and/or separate bus.

Rationale: Allows failed LP to be isolated from system by removal of power. How best to isolate failed LP's and meet the FO-S criteria is part of the overall study.

10.7.5.5 - Package the "power present" monitor with the load and route the signal over the LP/load interface. Redundant monitors may be desirable especially with redundant LP's. Rationale: Best place to sense power is where it is needed. Interface is already available. Sufficient redundancy is required in monitor to prevent monitor failure negating other redundancy measures.

10.7.5.6 - Determine required bus redundancy in conjunction with total required system redundancy. Rationale: Although parts of this study assumed four busses available to each load, the system redundancy requirements may possibly be met with one or two busses per load. This has to be determined from the overall system redundancy requirements.

11.0 RECOMMENDATIONS FOR FUTURE EFFORT

11.1 INTRODUCTION

A fault tolerant computer system capable of fail op, fail op, fail safe operation has been defined and the feasibility of the concept verified through software simulation. It is considered a significant step toward the goal of an operational fault tolerant computer system with application not only in the Space Station, but in other manned or unmanned space systems where high reliability is of utmost concern. The following list of activities are recommended as logical extensions of this study.

11.2 SOFTWARE AND SIMULATION

The simulation activity conducted during the study period provided primarily assurance of design feasibility and demonstration of the simulation system operations. There are two major areas where additional simulation activity would be particularly valuable.

11.2.1 VCS Fault Detection/Isolation

Since the voting functions in the VCS are the primary means of failure detection in the RGC system, it is particularly critical that the hardware/software system be mechanized in the manner that provides a high probability of proper isolation of VCS failures. The impact of erroneously assigning VCS failures to other system elements is very great. Therefore it would be highly desirable to investigate/evaluate system performance under a rather detailed simulation of VCS malfunctions. This can be accomplished by expanding the fault-generation capabilities in the Simulator and investigating a reasonable variety of VCS malfunction cases.

11.2.2 Module-level Reconfiguration

The RGC system is designed to allow for module-level (CPU, IOP, memory) reconfiguration within each of the two physical compartments. A preliminary analysis of the techniques required to mechanize the capability was performed during the study, but the necessary software routines were not programmed. System reliability, in terms of probability of mission success, can be significantly enhanced by adding the level of reconfiguration capability but the probability of successful reconfiguration must be high and the possibility of induced or propogated malfunctions caused by the additional reconfiguration paths must be essentially eliminated. Developing the required software routines and performing additional fault simulation would be valuable in assessing the feasibility/desirability of a module-level reconfiguration implementation.

11.2.2 (Continued)

Regardless of what specific follow-on simulation activity is planned, it would appear that a minimal effort should be expended on increased documentation and/or user-orientation on the Simulation System. This system represents a rather general tool which could be used for a variety of design evaluation tasks. Considering the investment already made, it would seem judicious to assume that full use of the tool can be made in the future.

11.3 COMPUTER/VCS STUDIES

Concurrent with the simulation activity, additional system studies should be conducted to investigate areas that were beyond the scope of this study. This effort should include the following:

- Further investigation of system fault tolerance, criticality of computations, fail op boundaries, etc.
- Further effort on design to guarantee failure independence to assure assumptions in meeting FOOS.
- VCS definition - Further study to permit designing VCS as an add on black box to today's off-the-shelf computers.
- Detailed logical design of VCS to permit a complete specification for hardware development.
- Further work on hierarchy of control in the selected computer organization, e.g., lockouts to assure a flexible design while still satisfying FOOS.
- Study of the applicability of the local processor design for other subsystems not covered under this study.

11.4 I/O DATA BUS STUDY

The baseline error protection scheme for the I/O data bus uses two-dimensional simple parity checks and request for retransmission. The performance of this error protection method depends on a large number of variables, many of which are difficult to estimate for the Space Station application. A more thorough evaluation of this protection method should be conducted. This can best be achieved by computer simulation, allowing a wide latitude of values for the variables to be exercised and simulating and computing the performance of the bus subsystem. Hardware tests are also necessary in this effort to help define the ranges of hardware dependent variables. Some of the variables to be exercised are: noise sources, probability distributions for

11.4 (Continued) noise, S/N ratios, data line lengths, message lengths, internal control word encoding, sample rates, message rates, and retransmission logics.

Various performance measures can be computed in the simulation such as throughput probability of undetected error(s) and IOP utilization. Given the simulation routine, the effects of changes to the baseline error protection scheme can be evaluated. These can include simplifying the scheme as well as increasing its complexity by, for example, appending a cyclic code check for burst error detection and/or correction. The performance for hardware or software forward error correction can also be computed. Combinations of coding techniques can also be evaluated, with the selected method at any point in time determined by subsystem message criticality, subsystems and system status, and system configuration.

12.0 REFERENCES

1-1 Detailed Program Plan, Reconfigurable G&C Computer Study Space Station Use, Autonetics C70-111/301.

4-1 Short, R. A., The Attainment of Reliable Digital Systems Through the Use of Redundancy - a survey, IEEE Computer Group News, March 1968.

4-2 Technical discussions with NASA - MSC.

4-3 Koczela, L. J., Study of Spaceborne Multiprocessing Phase I Final Report, NASA CR 1446, February 1970.

4-4 Koczela, L. J., The Distributed Processor Organization, Advances in Computers, Volume 9, Academic Press 1969.

4-5 Roth, J. P., Phase II of an Architectural Study for a Self-Repairing Computer, AD 825 460, November 1967.

4-6 Godberg, J., Techniques for the Realization of Ultrareliable Spaceborne Computers, N70-18784, June 1968.

4-7 Control, Guidance, and Navigation for Advanced Manned Missions; Volume II, Multiprocessor Computer Subsystem, MIT, N69-28660, January 1968.

4-8 Cok, F. B., Self Repair Techniques Investigation, AD 657 247, June 1967.

4-9 Agnew, P. W., An Architectural Study for a Self Repairing Computer, AD 474 976, November 1965.

5-1 Farr, D. L., et al: Spaceborne Computer Design Evaluation NASA Contract NAS 12-589, North American Rockwell Corp., Autonetics Division, July 1968.

5-2 Koczela, L. J., et al: Study of Spaceborne Multiprocessing Second Quarterly Report. NASA Contract NAS 12-108, North American Rockwell Corp., Autonetics Division, October 1966.

5-3 Anderson, M. D. and Marek, V. J.: Evaluation of Aerospace Computer Architecture. Paper No. 68-836, Presented at the AIAA Guidance, Control, and Flight Dynamics Conference, August 1968.

12.0 (continued)

5-4 Chestnut, Harold: Systems Engineering Tools, John Wiley & Sons, Inc., New York 1965.

5-5 Chapanis, Alphonse, et al: Operations Research and Systems Engineering, Johns Hopkins Press, Baltimore, 1960.

6-1 Abramson, N., and B. Elspas, Double Error Correcting Encoders and Decoders for Non-independent Binary Errors, UNESCO International Conference on Information Processing, 1959, pp 493-4.

6-2 Benice, R. J., and A. H. Frey, Comparisons of Error Control Techniques, IEEE Transactions on Communication Technology, December 1964, pp. 146-154.

6-3 Calingaert, P., Two-dimensional Parity Checking, Journal of the ACM, April 1961, pp 186-200.

6-4 Chien, R. T., Recent Developments in Algebraic Decoding, Proceedings International Telemetering Conference 1969.

6-5 Eisenbies, J. L., Conventions for Digital Data Communication Link Design, IBM System Journal, V.6 #4, 1967, pp 267-302.

6-6 Gallagher, R. G. Information Theory and Reliable Communication, Wiley 1968.

6-7 Hsiao, M. Y., and J. T. Tou, Application of Error-correcting Codes in Computer Reliability Studies, IEEE Transactions on Reliability, Vol. R-18 #3, August 1969, pp 108-118.

6-8 Kastenholz, C. E., Error Control Techniques in Tactical Command and Control Systems, Autonetics TM 341-2-4, May 4, 1964.

6-9 Kastenholz, C. E., General Purpose Computer Encoding and Decoding of Error Control Codes, National Electronics Conference, October 1964.

6-10 Lucky, R. W. and J. Salz, and E. J. Weldon, Principles of Data Communication, McGraw Hill, 1968.

6-11 Peterson, W. W., Error Correcting Codes, Wiley 1961.

6-12 Sellers, Hsiao, & Bearnson, Error Detecting Logic for Digital Computers, McGraw Hill 1968.

12.0 (continued)

6-13 Tang, D. T. and R. T. Chien, Coding for Error Control, IBM Systems Journal, #1, 1969, pp 48-86.

6-14 Wolf, J. K., et al:, Algebraic Coding and Digital Redundancy, IEEE Transactions on Reliability, Vol. R-18 #3, August 1969, pp 91-107.

6-15 Willard, M. W., Optimum Code Patterns for PCM Synchronization 1962 National Telemetering Conference Proceedings, May 1962, Paper #5-5.

10-1 SD 69-607 Systems Requirements Book (SRB) - Space Station Vol. II, Rev. E, Space Division, North American Rockwell Corp., February 13, 1970. NAS 9-9953.

10-2 Technical Interchange Meeting with NASA, January 21, 1970, at Autonetics.

10-3 Corey, P. D., Analytical Optimization of Magnetics for Static Power Conversion, Supplement to IEEE Transactions on Aerospace, June 1965, Vol. AS-3 No. 2, pp 86-92.

10-4 Grossner, N. R., Transformers for Electronic Circuits, McGraw Hill 1967.

10-5 Van De Riet, E. K., Feasibility Study for Reliable Magnetic Connection Switch, Final Report Phase II Contract 95-1232 under NAS 7-100 by Stanford Research Institute (SRI) for Jet Propulsion Laboratories (JPL).

10-6 NR Corp. Laboratory/Engineering Notebook N7528, pp 25-28, Magnetic Amplifier Calculation.

10-7 LTV Aerospace Corp., Vought Aeronautics Division, Dallas, Texas, Procurement Specification for Power Controllers, Rev. D, 10-15-69.

10-8 Heinemann Bulletin 3504 Series SM Subminiature Circuit Breakers.

10-9 Warner, M., Jr., Integrated Circuits Design Principles and Fabrication, McGraw Hill 1965.

10-10 Yu, A.Y.C., The Metal-Semiconductor Contact: An Old Device with a New Future, IEEE Spectrum, March 1970, pp 83-89.

10-11 Gibbons, J. F., Ion Implantation in Semiconductors - Part I Range Distribution Theory and Experiments, IEEE Proc. Vol. 56, No. 3, March 1968, pp 295-319.